NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

EVALUATION OF THE USE OF GPS-AIDED WEAPONS TO ATTACK MOVING TARGETS

by

Randolph L. Mahr

March 2001

Thesis Advisor:	Russell Duren
Second Reader:	Morris Driels

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** March 2001	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE: Title (Mix case letters) Evaluation of the Use of GPS-Aided Weapons to Attack Moving Targets			**5. FUNDING NUMBERS**
6. AUTHOR(S) CDR Randolph L. Mahr, USN			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			**12b. DISTRIBUTION CODE**

13. ABSTRACT ***(maximum 200 words)***

The current intelligence gathering and strike decision infrastructure is optimized to handle geographically and temporally fixed targets. When tasked to respond to targets that require near immediate engagement, however, the system is stressed to the limit of its capability. When these Time Sensitive Targets are capable of relocating, the process of rapidly applying lethal force becomes even more complicated. This thesis examines the problems associated with attacking a moving target using low cost GPS-aided standoff weapons, without an integrated weapon seeker. It begins with a discussion of the history and evolution of the Navy's ability to attack time sensitive moving targets, and provides the description of a system that could address shortcomings noted. MATLAB® Simulink® was used to develop a model to simulate the proposed system, and determine the responses to various combinations of identified error sources. The results of the research showed that the type of system proposed is technically feasible.

14. SUBJECT TERMS GPS, Weapon, Modeling, CEP, Time Sensitive Targets, Command Control and Communications, Conventional Weapons, Sensors, Modeling and Simulation, Time Critical Strike			**15. NUMBER OF PAGES** 172
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited.

EVALUATION OF THE USE OF GPS-AIDED WEAPONS TO ATTACK MOVING TARGETS

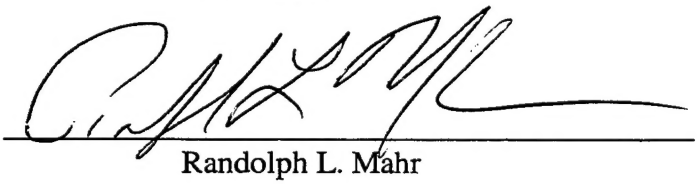
Randolph L. Mahr
Commander, United States Navy
B.S., United States Naval Academy, 1983

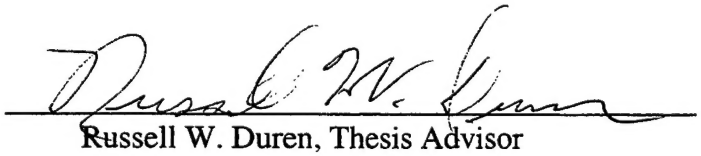
Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN AERONAUTICAL ENGINEERING

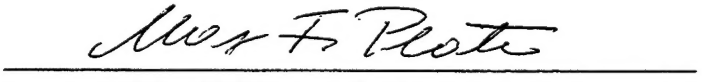
from the

NAVAL POSTGRADUATE SCHOOL
March 2001

Author: 
Randolph L. Mahr

Approved by: 
Russell W. Duren, Thesis Advisor


Morris R. Driels, Second Reader


Max F. Platzer, Chairman
Department of Aeronautics and Astronautics

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The current intelligence gathering and strike decision infrastructure is optimized to handle geographically and temporally fixed targets. When tasked to respond to targets that require near immediate engagement, however, the system is stressed to the limit of its capability. When these Time Sensitive Targets are capable of relocating, the process of rapidly applying lethal force becomes even more complicated. This thesis examines the problems associated with attacking a moving target using low cost GPS-aided standoff weapons, without an integrated weapon seeker. It begins with a discussion of the history and evolution of the Navy's ability to attack time sensitive moving targets, and provides the description of a system that could address shortcomings noted. MATLAB® Simulink® was used to develop a model to simulate the proposed system, and determine the responses to various combinations of identified error sources. The results of the research showed that the type of system proposed is technically feasible.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

PREFACE 1

I. INTRODUCTION 3
- A. EVOLUTION OF NAVAL AIR-AIR WARFARE 3
- B. EVOLUTION OF NAVAL STRIKE WARFARE 3
  - 1. Strike Warfare Master Plan 4
  - 2. Weapon Development 5
  - 3. Joint Standoff Weapon (JSOW) 5
- C. STRIKE WARFARE PROCESS 6
  - 1. Responsive Targeting 7
  - 2. GPS Weapons Against Time Sensitive Moving Targets 8
  - 3. Addressing Deficiencies 8
  - 4. Time Sensitive Moving Target System 8

II. SYSTEM DESCRIPTION 11
- A. IDEAL SYSTEM DESCRIPTION 11
- B. PROPOSED SYSTEM 11
  - 1. System Context 13
  - 2. System Overview 14
    - *a. Sensor Link 15*
    - *b. Data Processing and Fusion System 16*
    - *c. Data Transmission System 16*
    - *d. Weapon 17*
    - *e. Additional Terms 17*

III. MODEL DESCRIPTION 19
- A. OVERVIEW 19
- B. TARGET TRUTH SUBSYSTEM 21
  - 1. Road Movement 22
  - 2. Constrained Random Motion 24
- C. SENSOR/TRACKER SUBSYSTEM 26
  - 1. Image Processor Subsystem 27
  - 2. Road Map Subsystem 30
  - 3. Movement Prediction 33
  - 4. Data Message Format 37
  - 5. Target DR Subsystem 39
- D. TRANSMISSION SUBSYSTEM 39
- E. MISSILE BEHAVIOR SUBSYSTEM 40
- F. INPUT AND ANALYSIS SUBSYSTEMS 44
  - 1. System Input and Errors Subsystem 44
  - 2. Display Subsystem 44
  - 3. Lethality Subsystem 45

IV. SIMULATION RESULTS 47
A. OVERVIEW 47
1. Movement Prediction 47
2. Example of Intercept Geometry 47
3. Predicted Impact Points 49
4. Calculation of Radial Miss Distance 50
B. DATA ACQUISITION PLAN 51
C. WHAT DO THE RESULTS MEAN? 52
D. DATA ANALYSIS 55
1. Perfect System 55
2. Relative Effect of IPI Latency 56
3. Variable Image Processing Interval 57
4. Imagery Induced Errors 58
5. Relative Impact of TLE and TVE 59
6. Variable Target Location Error 60
7. Variable Data Transmission Interval 61
8. Effectiveness Against a Constrained Random Motion Target 63
E. DATA VALIDITY 64
1. Missile Behavior 64
2. Statistical Analysis 66
3. Sample Size 72
4. Gross Errors 72
V. CONCLUSIONS AND RECOMMENDATIONS 75
A. SUMMARY OF RESULTS 75
B. TACTICAL CONSIDERATION 76
C. RECOMMENDATIONS FOR FUTURE RESEARCH 76
APPENDIX A. MODEL USE 79
A. SET UP 79
B. DEFAULT MODEL USE 79
C. MODIFYING DEFAULT VALUES 79
D. INTERNAL CHANGES 81
E. MULTIPLE SEQUENTIAL RUNS 83
APPENDIX B. SIMULINK® MODELS 85
APPENDIX C. MATLAB® M-FILES 103
APPENDIX D. TARGET TRUE MOTION MATLAB FILES 129
APPENDIX E. TABULATED RESULTS 135
LIST OF REFERENCES 145
SELECTED BIBLIOGRAPHY 147
GLOSSARY 149
INITIAL DISTRIBUTION LIST 153

LIST OF FIGURES

Figure 1. System Context Diagram 14
Figure 2. TSMTS System Overview 15
Figure 3. Relationship of TSMTS and Simulink® Model 20
Figure 4. Top Level Simulink® Model 21
Figure 5. Target Truth Subsystem 22
Figure 6. Road Map for Simulated Target Travel 23
Figure 7. Variable Velocity Profile for Road A 24
Figure 8. Variable Velocity Profile for Road B 24
Figure 9. Histogram of Typical Velocity Change Inputs 25
Figure 10. Typical 2-D Path for Target with Constrained Motion 26
Figure 11. Sensor/Tracker Subsystem 27
Figure 12. Adding Error to Target Truth Data 27
Figure 13. Typical Position Error Inputs for One Simulation Run 28
Figure 14. Typical Velocity Error Inputs for One Simulation Run 28
Figure 15. Example of Variable Image Latency 29
Figure 16. Estimated Target Position 31
Figure 17. Predicted Target Position after 1 Time Step 32
Figure 18. Predicting Target Position Beyond One Turn Point 33
Figure 19. Effect of 1-Knot Constant Error 36
Figure 20. Difference Between Predicted Positions 37
Figure 21. Transmission Subsystem 39
Figure 22. Example of Cascading Latency Effects 40
Figure 23. Missile Behavior Subsystem 43
Figure 24. Calculating Radial Miss Distance 45
Figure 25. Predicted Target Motion Points 48
Figure 26. Missile Prediction of Intercept, Road B with variable Velocity 49
Figure 27. Predicted Impact Points 49
Figure 28. Expanded View of Predicted Impact Points 50
Figure 29. Example of Radial Miss Distance 51
Figure 30. Effect of Increasing CEP_{TLE} 54
Figure 31. CEP Due to Image Process Time 57
Figure 32. CEP Due to Image Process Time 57
Figure 33. Relative Impact of Variable IPI 58
Figure 34. CEP Due to Target Location and Velocity Errors 59
Figure 35. Relative Effects of TLE and TVE on CEP 60
Figure 36. Variable TLE with constant IPI 61
Figure 37. Variable TLE with constant IPI 61
Figure 38. Variable Data Transmission Interval 62
Figure 39. Variable Data Transmission Interval 63
Figure 40. Effectiveness Against Random Target Motion 64
Figure 41. Typical Weapon Track to Intercept without Targeting Error 65

Figure 42. Typical Weapon Track to Intercept with Targeting Error 65
Figure 43. Typical Track to Intercept with Targeting Error 66
Figure 44. Resolving RMD into Range and Deflection Components 67
Figure 45. Range Error Distribution 68
Figure 46. Graph of Distributions from Table 6. 71
Figure 47. Sample Size Comparison 72
Figure 48. Percentage of Type-2 Gross Errors by Road Profile 73
Figure 49. CEP_{Total} with Increasing CEP_{Weapon} and Fixed CEP_{TLE} 76
Figure A-1. Primary Input Variables 81

LIST OF TABLES

Table 1. Simulation Variables 52
Table 2. Variable Combinations Used During Data Collection 56
Table 3. Histogram of Raw Range and Deflection Data 68
Table 4. Histogram of Range and Deflection Data with 'Direct Hits' Removed ... 69
Table 5. Probability Analysis of Raw Data 70
Table 6. Histogram Comparison of Distributions 71
Table 7. Critical Technical Parameters 75

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

The author would like to acknowledge the support of the Program Executive Office (Strike Weapons and Unmanned Aviation), Conventional Strike Weapons Program Office (PMA-201). Specifically, Mr. Keith Sanders, CDR Aaron Bowman, Mr. Rex Roberts, and Ms. Maryanne Martin provided support during the research phase of this thesis. CDR Tom Glass USN (ret.), currently employed by the Computer Sciences Corporation and contracted to support PMA-201, was especially helpful in collection of diverse sources and providing materials otherwise unobtainable.

The author would also like to thank Dr. Russ Duren and Dr. Morris Driels for their support, insight and encouragement.

THIS PAGE INTENTIONALLY LEFT BLANK

PREFACE

This thesis presents the results of research conducted in partial fulfillment of the requirements for a Masters of Science Degree in Aeronautical Engineering from the Naval Postgraduate School. The thesis is a summary of the rationale for the particular concerns unique to attacking moving targets with GPS-aided weapons, and the course of analysis and experimentation that was conducted to proffer a solution.

Chapter I discusses the history and evolution of the Navy's ability to attack time sensitive moving targets.

Chapter II provides the description of a system that could address the shortcomings noted in Chapter I, and identifies the error sources associated with each component of the system.

Chapter III describes the particular model developed specifically for this research to compare the results of selected variables on the utility of the system.

Chapter IV presents the results of the simulations and supporting analysis.

Chapter V summarizes the findings, and makes recommendations for future efforts.

Appendices A through D contain the supporting source code and diagrams used during the analysis. Appendix E contains tabulated results of the data from the simulations.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. EVOLUTION OF NAVAL AIR-AIR WARFARE

In the late 1960's the Navy fielded a formidable air-air capability in Southeast Asia. The F-4 Phantom II, coupled with the very capable Sidewinder and Sparrow missiles, proved a deadly threat to the North Vietnamese Air Force, as well as a credible deterrent to the Cold War foes of United States policy. Unfortunately, to engage the enemy, the crew of the venerable Phantom had to depend on their own visual acuity and the limited range of their on-board radar. While they were equipped with increasingly capable weapons, they had to operate in a very dynamic environment with only occasional cueing, mostly limited to UHF radio transmissions of general enemy locations. The capabilities of their weapons far outstripped the ability of the intelligence system to distribute useful data, denying their Commanders a large percentage of potential combat effectiveness.

By 1990, when the next major air-air conflict occurred over Iraq, not only had aircraft and weapons been modernized, but also a theater-wide picture of the air war was now available to combat Commanders. They had access to an integrated network which allowed them to evaluate information about potentially hostile aircraft and missiles, direct essential information to airborne Command, Control, Communications and Intelligence (C3I) platforms, and then to the fighters via dedicated data links. The pilots in the fighters were able to quickly sort the target data in a useable way between aircraft, and employ their weapons very efficiently. This networked system for obtaining and distributing information greatly increased the lethality of the coalition air forces. More importantly it allowed them to make use of a greater portion of the capability inherent in the integrated weapon system, improving the efficiency with which the air war was conducted.

B. EVOLUTION OF NAVAL STRIKE WARFARE

While integrated air-air warfare matured, the air-ground system made fewer advances. In the Gulf War, F/A-18 pilots were in a situation similar to that which the Phantom crew faced in Southeast Asia: the system could not quickly produce the

necessary targeting and weaponeering information to fully utilize the capability of their weapon system in a dynamic scenario, and when it was produced, the system could not distribute the information to the shooter in a timely manner.

1. Strike Warfare Master Plan

Slightly more than a decade ago the Navy's Strike Warfare Master Plan (SWMP) was developed[1]. The plan reflected the lessons learned in combat and strike training since the end of the Vietnam War, relied heavily on naval aircraft with two crewmembers, and postulated scenarios that depended on Man-In-The-Loop (MITL) techniques to get weapons on the right target. There was no concern for joint service operations, and almost no consideration of system enhancements required to utilize the increasing weapon capabilities.

No sooner had the ink dried on the original plan than the Soviet Union began to fracture, and concurrently Desert Storm tested the veracity of the lessons learned, and the planning which they yielded. Commanders were abruptly faced with forced involvement in joint service actions that required them to provide better strike force coordination and deconfliction of targets. These new issues affected weapon selection, as well as the problems of integrating service-specific ('stovepiped') targeting and delivery systems that had until that time been able to operate largely independently. Across the services, political objectives in limited hostilities forced Commanders to try to minimize the probability of aircraft attrition, while delivering weapons with more precision to lower the risk of collateral damage. At the same time Naval Aviation retired the A-6 Intruder, the aging two-seat strike workhorse which had allowed a dedicated B/N to concentrate on weapon delivery for long periods while the pilot flew the airplane. The resulting dependence on the F/A-18 as the primary carrier-based strike aircraft caused the Navy to reconsider the complexity of the system of weapon systems it fielded, in an attempt to bring integrated tactical flexibility, reliability and multiple kills per pass to the warfighting Commander.

---

[1] The "*Strike Warfare Master Plan*" has not been updated, and was unavailable for review. The author of this paper is familiar with its contents from previous assignments in weapon acquisition at Naval Air Systems Command.

2. Weapon Development

To support the roadmap laid down by the SWMP, new weapon systems were being procured. The Joint Standoff Weapon (JSOW), Joint Direct Attack Munition (JDAM) and Standoff Land Attack Missile (SLAM) all entered service shortly before the turn of the century. One enhanced capability these weapons brought to the Fleet was improved accuracy from longer ranges resulting from the integration of the Global Positioning System (GPS) with their inertial guidance systems. This standoff allowed the launch aircraft to stay out of the range of many threat systems, reducing the likelihood of aircraft attrition. The GPS-aided navigation also provided enough accuracy to allow the Navy to move away from MITL control and reduce pilot workload.

3. Joint Standoff Weapon (JSOW)

The evolution of the JSOW Unitary (AGM-154C) program is an excellent example of the changes Strike Warfare weapon development has undergone. Conceived as the Advanced Interdiction Weapon System (AIWS) program in 1986, it was designed to provide a broad capability against a spectrum of well-defined point targets, including design consideration of moving targets such as trains and armed naval combatants.[2] Its design incorporated a state-of-the-art seeker, coupled to a digital data-link to enable the pilot to guide the weapon to very precise point of impact. Despite the end of the Cold War and resulting decreases in the budget, JSOW Unitary was still listed as the Fleet's No. 1 priority at Weapons Operational Advisory Groups as late as 1998.[3] However, the budget belt was tightening, and although the need was still real, the capability was too expensive. The Resource Sponsor, N88, challenged the program to reduce the production price. To achieve affordability they allowed the Program Office to consider modifying any requirement other than those defined as key performance parameters (KPP).[4]

In response the program began a Cost as an Independent Variable (CAIV) effort. The JSOW contractor, Raytheon, considered modifications to the original configuration

---

[2] JSOW Operational Requirements Document (ORD).

[3] PMA-201, JSOW Unitary Program briefing materials.

[4] Ibid.

and found that 95% of the requirement could be met for 60% of the cost. The key to this cost reduction involved removing the data link and data link antennas, and reducing the seeker window in size. The resulting savings of approximately $125,000 per unit addressed the affordability issues at that time.[5]

Two good things happened when the data link was eliminated: pilot workload was significantly reduced, and no data link pod was required – which opened up a valuable weapon station. But, nothing is free. With the data link went all ability to communicate with the weapon after launch, which eliminated the opportunity for post launch re-targeting of the weapon against the portion of the target list that included moving targets.

C. STRIKE WARFARE PROCESS

Warfare in the latter half of the 20th century has become a holistic effort. The battlefield is looked at as a whole, and forces are applied where they can be most effective based on assessment of the tactical and strategic situation. While JSOW and other weapons have helped improve the 'hardware' available to the fleet, Commanders have encountered limitations in the 'process' of waging war using these weapons.

Much as the Phantom crew in Vietnam operated using only a portion of their full capability, the Strike/Fighter pilot in Southwest Asia today must go to war dependant on a system that provides only a limited amount of temporally 'stale' ground target information. After launching from the carrier, today's pilot receives almost no outside assistance in locating or prioritizing targets in the dynamic environment. In most cases information can only be received via voice radio, which requires the pilot to comprehend a complex, dynamic battlefield situation, and manually modify weapon attack parameters to employ the weapons. There is no way to provide updated target information after a weapon is launched. With these restrictions, when faced with a mobile or moving target, the pilot has little possibility of being able to use the full capability of the weapons with which he or she is equipped since most current strike aircraft sensors don't have the

[5] PMA-201, JSOW Unitary Program briefing materials.

accuracy necessary at significant standoff ranges to support dynamically targeting GPS weapons.

1. Responsive Targeting

The current intelligence gathering and strike decision infrastructure is optimized to handle geographically and temporally fixed targets. Stripping the system to its base, it is obvious the fidelity of available information, system interoperability, system connectivity, and robustness of the interfaces will regulate the targeting cycle speed. Stationary targets, such as buildings or bridges, allow the system sufficient time to complete the entire targeting cycle from detection to assessment of strike effectiveness. However, when tasked to respond to targets that require near immediate engagement, the system is stressed to the limit of its capability[6]. This class of target has been termed a Time Sensitive Target (TST)[7].

Immediate warfighter importance, and compressed vulnerability windows characterize TSTs. They span the tactical, operational, and strategic target set, they can be found throughout the battlefield, and they must be engaged under a wide range of operating conditions. It is true that using an accelerated decision making process and rapid application of dedicated on-call strike forces can successfully attack some TSTs. This tactic was used with limited success in Desert Storm when aircraft were assigned to combat air patrol (CAP) points from which they could be called in to attack SCUD missiles that were located by intelligence assets. However, from a weapon availability perspective this tactic is extremely inefficient, since it severely limits the utilization of the aircraft when in this CAP role. Striking a TST becomes even more difficult when the target is capable of relocating, combined with self-defense, or while protected by overlapping air-defense systems. It is desirable to find an affordable means of employing currently fielded conventional standoff weapons against a target that has the capability of

---

[6] Jewett, "Making Network Centric Warfare Real", briefing materials, 18 Jan 00.

[7] "Time-sensitive targets – those targets requiring immediate response because they pose (or will soon pose) a clear and present danger to friendly forces or are highly lucrative, fleeting targets of opportunity.", Joint Publication 1-02, *DoD Dictionary of Military and Associated Terms* as reprinted in U.S. Joint Forces Command Joint Warfighting Center, *A Common Perspective*, Vol.8, No. 2, October 2000, pp7.

geographically moving within the targeting cycle, while maintaining tactical flexibility for the Commander.

2. GPS Weapons Against Time Sensitive Moving Targets

The current generation of low-cost standoff weapons (including JSOW, JDAM, and SLAM-ER) rely heavily on the Global Positioning System (GPS) to guide themselves to targets located at fixed geo-spatial coordinates which are loaded in the weapon prior to release from the launch platform. When attacking a fixed site GPS-aided guidance can provide sufficient accuracy to allow the elimination of costly seeker-based guidance systems, as was the case with JSOW Unitary. However, GPS-only guidance is ineffective if the target is mobile or moving, since the weapon is navigating to a fixed point in the geospace grid. Even if a seeker were placed on the weapon, it would still be limited by the field-of-regard of the seeker. So, although these weapons are fielded, and the intelligence system can find the time sensitive moving target (TSMT), there is a lack of system capability to attack a TSMT with precision standoff weapons

3. Addressing Deficiencies

The growth in the networked intelligence and force application of the air-air arena provides a model for a system to address this deficiency. Taking the best capabilities developed for that scenario, including data collection, continuous target tracking, and information distribution, and providing them to the strike pilot will make a step improvement in strike capability. Most importantly, this can be accomplished within existing technological boundaries.

Using the air-air model, the challenge is to connect the existing system of sensors which can track ground moving targets, and using an information processing station develop and continually supply updated target position to weapons. To some extent this technique is analogous to the 'command line-of-sight beam-rider' technology in air-to-air or surface-to-air missiles, the principle difference being the lack of any direct communication link between the sensors and the weapon.

4. Time Sensitive Moving Target System

Naval Aviation is rapidly approaching its' 100th birthday. The past century has seen a progression from battleships launching biplanes with box fin bombs to highly

capable supersonic strike fighters carrying weapons which depend on orbital satellites for guidance. This thesis will develop a requirements-based model of a sensor-to-weapon system to provide constantly updated target location information to a GPS-guided weapon after launch.

THIS PAGE INTENTIONALLY LEFT BLANK

II. SYSTEM DESCRIPTION

A. IDEAL SYSTEM DESCRIPTION

Attacking a time sensitive moving target using a GPS-aided conventional standoff weapon does not necessarily require the development of a purpose-built weapon. Rather, it can be accomplished by using predominantly existing technologies integrated into a system of weapon systems. To accomplish this, the task of the system design engineer is to define a continuous path for formatted information about target behavior to be passed from a sensor to a weapon. Thus, from a systems engineering perspective, the sensor will provide input into the system, and the output will be the effect on the target.

Looking at the 'ideal' system, some key behaviors may be described by considering the interfaces across the boundaries of the system at the sensor 'input', and the target 'output'. An ideal system should be flexible enough to accept data from a variety of sensors, and the resulting capability to attack a target should be unaffected by the source. An ideal system should be able to interface with any of a variety of currently fielded weapons, providing data that is compatible with individual guidance and control systems. The behavior of the ideal system within these boundaries may be described as:

> *When a moving target of interest is identified, and the command to attack that target is given, the system will provide continually updated, perfectly accurate target location, velocity and acceleration vector data to a weapon. This data will be provided at appropriate intervals to the weapon, from pre-launch to target impact to enable the weapon to adjust it's flight path to allow target interception. The probability of killing the moving target will not be degraded from the probability of killing the same target if it were stationary.*

B. PROPOSED SYSTEM

A proposed Time Sensitive Moving Target System (TSMTS) can be viewed in many ways to evaluate the ability of any solution to approach the ideal. For this research, a functional decomposition based on real-time modeling techniques has been chosen.

Derek J. Hatley and Imtiaz A. Pirbhai developed the best known of this type of system specification in the 1980s, when they proposed a model based on their experience at a major avionics systems development company. They were faced with designing and testing increasingly complex avionics systems which existing developmental methods were unable to handle satisfactorily. Hatley and Pirbhai realized that one cause of their problem was that specification methods available to them addressed only one, or at best a very few, aspects of system design. They knew that in reality complex systems have many aspects that need to be addressed during the design phase. One of their key realizations was: there comes a point in system development "at which the interactions between the subsystems are at least as complex as the subsystems themselves."[8]

The Hatley-Pirbhai Model approached these problems by providing a formal methodology to help define the system based on functional boundaries, and by defining multiple ways for the Systems Engineer and System Architect to view the system design. In a Hatley-Pirbhai 'functional decomposition' the system is viewed as a set of interactive functional components or activities organized into a hierarchy. This view of a system encouraged the designers to consider each function, and where applicable further subdivide each until the entire system was specified in terms of basic activities.[9] Importantly, the model of the system was largely independent of the planned physical implementation of the design.

Hatley and Pirbhai also realized that a useful model needed to address "human readability and understandability through the use of graphics."[10] Capitalizing on the popularity of the Hatley-Pirbhai model with the commercial aircraft and automobile manufacturing industries[11], several software vendors have developed 'tools' that

---

[8] Hatley and Pirbhai, *Strategies for Real-Time System Specification*, p.5.

[9] A 'basic activity' is one that the system design team decides requires no further decomposition.

[10] Hatley and Pirbhai, *Strategies for Real-Time System Specification*, p.5.

[11] Hatley and Pirbhai successfully applied their technique to various design problems, and state that this methodology directly enabled the Federal Aviation Administration to certify a very complex real-time embedded avionics system for a

automate the development of specifications using the functional decomposition methodology. After evaluating several of these, "Statemate MAGNUM"™ [12] was initially selected for use in this research to quickly synthesize the system into an assessable form. Previous versions of this software have been successfully used by the Naval Air Systems Command to develop Avionics System Specifications. However, after initial work was completed on the top-level design, several training issues prevented completion and implementation using that software, therefore the final model was completed and data developed using MATLAB® and Simulink®.[13]

1. System Context

Every system may be viewed as existing within another system's environment. The environment provides the 'truth' that the system estimates and reacts to, and is referred to as the 'context' of the system by Hatley and Pirbhai. The environment of the TSMTS is shown in Figure 1. The external activities, which affect the TSMTS, are the Decision-Maker, the Sensor, and the GPS System. Note the target is neither part of the TSMTS, nor does it interact directly with the TSMTS. From the perspective of TSMTS, the Target interacts only with the Sensor, therefore is not explicitly modeled.

---

commercial airliner in a relatively short time, and has since become widely adopted in the commercial aircraft industry.

[12] Statemate MAGNUM is a registered trademark of I-Logix Inc. Use of Statemate MAGNUM software for this project is under license to the Naval Postgraduate School.

[13] MATLAB and Simulink are registered trademarks of The MathWorks, Inc.

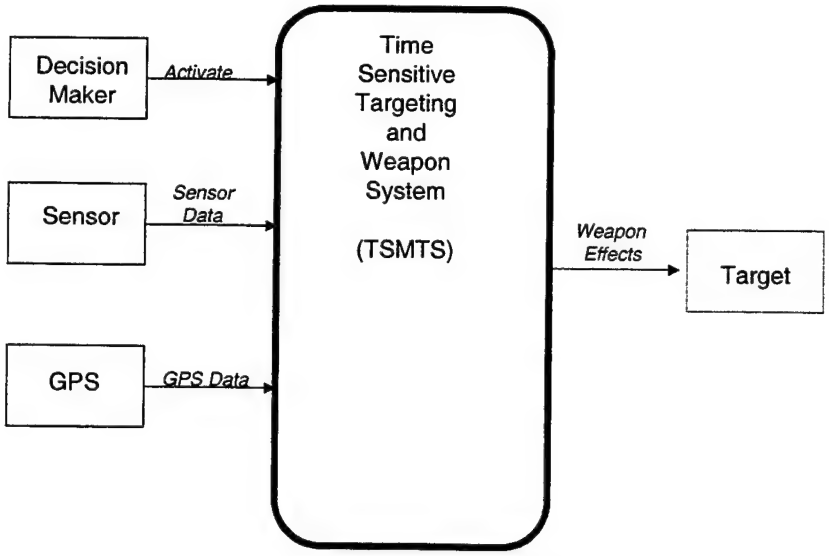


Figure 1. System Context Diagram

2. System Overview

TSMTS can best be described as a 'reactive system', one that exhibits the following general characteristics of such a system[14]:

- It continuously interacts with its environment. Inputs and outputs are often asynchronous, and may be either continuous or time discrete.
- It must be able to respond to interrupts
- Its operation and reaction to inputs reflect stringent time requirements
- It has many possible scenarios of operation, depending on the current mode of operation, current values of data, as well as past behavior.
- It is based on interacting processes that operate in parallel.

Moving from the contextual view into the environment of the TSMTS, the system can be functionally decomposed as shown in Figure 2. As shown, there are four principal parts to TSMTS: a Link from the Sensor, a Data Processing and Fusion System, a Data Transmission System, and a Weapon. Each of these parts is, of course, a highly complex system by itself. The challenge is to develop a model of the TSMTS that adequately

[14] Harel and Pollti, *Modeling Reactive Systems with Statecharts: The Statemate Approach*, pp. 1-4.

represents behavior of the interaction of multiple complex systems, without requiring excessive computer resources to run and evaluate.

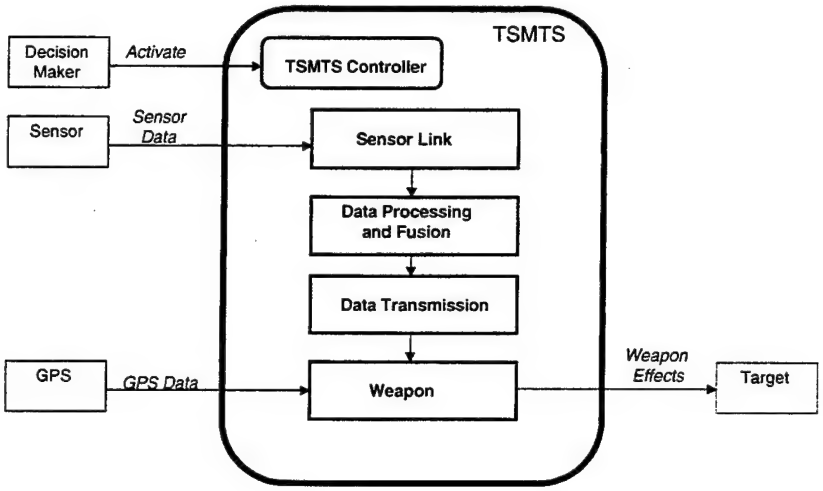


Figure 2. TSMTS System Overview

a. Sensor Link

The Sensor is external to the TSMTS, and interfaces with the TSMTS through a unidirectional data-link. The Sensor is a tracker which has the ability to 'look' at a geographic region when directed by its controller, as well as self-locate relative to the earth, however it does not necessarily track a target or determine absolute target location in the field of view. Spectral range of operation of the Sensor is not important to the TSMTS. Examples of Sensors include satellites and high-resolution airborne radar such as JSTARS. The Sensor has the capability to transmit scene information around the target of interest that will allow a ground station to register the imagery and geo-locate the target. The transmitted information comes into the TSMTS periodically via a dedicated data-link with a Sensor generated time-tag to document origination time. Errors associated with the data from the Sensor are: spatial and specific resolution of the target and target dimensions, the errors associated with self-location, the effects of atmospheric distortion on the image and image location, and the accuracy of the time associated with each image frame.

b. Data Processing and Fusion System

The Data Processing and Fusion System (DP/FS) is a ground-based system that is composed of two subsystems: Data Processing Station (DPS) and Data Fusion Cell (DFC).

The Data Processing Station is a high-speed computer system operated by skilled personnel which receives information from the Sensor, identifies the target within the context of the scene, and provides the notated image to the Data Fusion Cell. When the data is returned from the DFC, the DPS uses an algorithm based on track historical data and any known terrain or man-made features to predict target movement. It then formats the track information to provide data in a usable form to the transmission system. Data output to the transmission system is aperiodic. Errors associated with the DPS include processing errors due to incorrect target track designation, time latency due to processing and incorrect application of target movement prediction algorithms.

The Data Fusion Cell is a high-speed computer system that has a database of registered imagery appropriate for the area of interest. The data received from the DPS is matched through a registration process to the archived imagery, and the coordinates of the target in that frame are extracted. Errors associated with the DFC include errors inherent in the data base information, errors in the registration process, errors associated with tracking the centroid of a target, errors in developing the mensurated target coordinates, and inherent datum errors.

c. Data Transmission System

The Data Transmission System (DTS) includes a ground-based transmitter that receives formatted information from the Data Processing and Fusion System. The DTS may include multiple relay nodes that retransmit the formatted data to the weapon. Data is transmitted and maintained digitally in and between nodes. Data transmission times are periodic and discrete, allowing for time multiplexing of single data-link frequencies, if necessary[15]. Errors associated with the transmission system include

---

[15] Link-16 is a tactical data link specified in MIL-STD-6016 in use by the United States which provides high speed, secure transmission among several users. One possible method of providing data to a weapon would be through a Link-16 receiver on a weapon.

antenna losses into and out of each transmission node, atmospheric attenuation, background noise, and latency. (Intentional jamming of the DTS will not be addressed.)

d. Weapon

The Weapon is an unmanned self-maneuvering air vehicle capable of adjusting its flight path based on present position, environmental factors, and target coordinates loaded into its navigation system. The Weapon uses information from to develop its own navigation solution, and maintains that solution relative to the World Geodetic Survey 1984 (WGS-84) ellipsoidal model. Due to survivability concerns, the Weapon does not communicate its own location to any other part of the system. The Weapon contains an algorithm to predict the intercept point with the target based on Weapon present position and target predicted movement received from the DTS. Errors associated with the Weapon include self-location errors due to the accuracy of the onboard navigation system and maneuvering capability, target movement prediction, and terminal accuracy with either seeker or by weapon mean area of effectiveness.

e. Additional Terms

Three additional terms are useful when describing the System.

(1) Truth. Truth is the actual location and time in the physical world. The TSMTS can only estimate truth.

(2) Target. The Target is a vehicle moving on the ground that has an instantaneous location, and a velocity and an acceleration vector to control future position. The Target is external to the TSMTS and therefore always in Truth.

(3) Ground. Ground is the physical terrain feature that interacts with the Target and may obscure the Target from view of the Sensor.

THIS PAGE INTENTIONALLY LEFT BLANK

III. MODEL DESCRIPTION

A. OVERVIEW

The purpose of the model developed for this research was to assess requirements for GPS-aided low cost conventional standoff weapon to be used to attack time sensitive moving targets without the aid of a weapon-based terminal seeker. Since no 'real' system exists, the model makes reasonable approximations of the top-level behavior expected of the proposed system and the analysis attempts to determine requirement boundaries for satisfactory operation of such a system. Detailed definitions for model components were limited to that fidelity necessary to achieve the objective. The model does not attempt to produce absolute precision; rather it should be used to assess the relative measure of 'goodness'. The model contains representations of the principal error sources described in Chapter II and the output can be used to analyze various combinations of critical inputs[16].

The model was developed using the MATLAB® Simulink® program[17] as a simplified representation of the proposed real-world system. The model assumes that a target has been identified, a track established, and the decision to attack with an appropriate GPS-aided weapon has been made. Primary input to the model is target location and motion. The simulation begins at the release point of the weapon from the launch aircraft, and ends with the determination of a radial miss distance that is principally composed of target location error.[18] Operating instructions for the model can be found in Appendix A.

---

[16] For ease of use in the NPS computing environment, all values and performance data, including input and output of this model, is unclassified, and no contractor proprietary data was used to develop this model.

[17] MATLAB version 5.3.0.10183 (R11) was used under license of the Naval Postgraduate School.

[18] Target Location Error (TLE) is the difference between where the targeting system or weapon calculates the target to be, and the actual location of the target in the physical world. TLE has both a horizontal and a vertical component, however, this

The model was specifically designed with modular components that will allow for easier incorporation of upgrades in the future, should this development be continued. Within the model the system is partitioned by functionality to mirror the proposed system described in Chapter II, with the main functional objects being the "Target Truth Subsystem", the "Sensor/Tracker Subsystem", the "Transmission Subsystem", and the "Missile Behavior Subsystem." For data input and analysis three additional subsystem components were added: "System Inputs and Errors", "Display Subsystem" and "Lethality Subsystem." Figure 3 shows these modular objects in relation to the system described in the previous chapter. For the remainder of this document the model component names will be used. Figure 4 provides the top-level Simulink® diagram of the system model. Details of each subsystem block can be found in Appendix B.

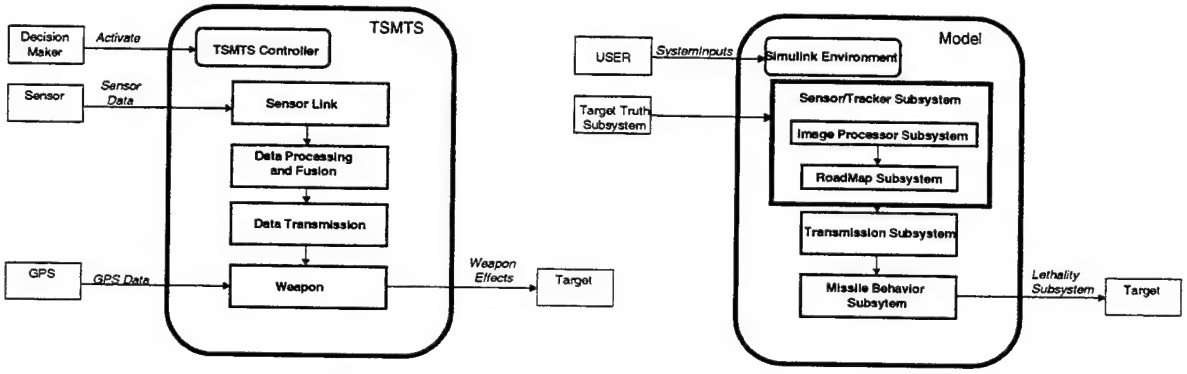


Figure 3. Relationship of TSMTS and Simulink® Model

All motion in the model is limited to a two-dimensional (2-D) plane, with the target start point always at the origin (0,0)[19]. The weapon starts at a random point at least 5 NM but not more than approximately 35.4 NM away from the origin. Starting farther from the origin impacted only time of flight, all dynamics of the weapon-target system level interaction can occur within this radius. The simplification to limit motion to 2-D

---

simulation only deals with the horizontal plane. TLE sources can include mensuration error from imagery, datum, datum transformation, and datum registration errors, equipment operator errors, and in the case of moving targets, predictive algorithm errors. The choice of a particular weapon or delivery mode has no effect on TLE.

[19] A limitation of the model requires that the x- and y-coordinates of the road definition be monotonically increasing relative to the origin for the duration of the simulation.

allows consideration of multiple missile types, each of which may implement their own altitude control methodology, but this simplification does impact absolute model accuracy.

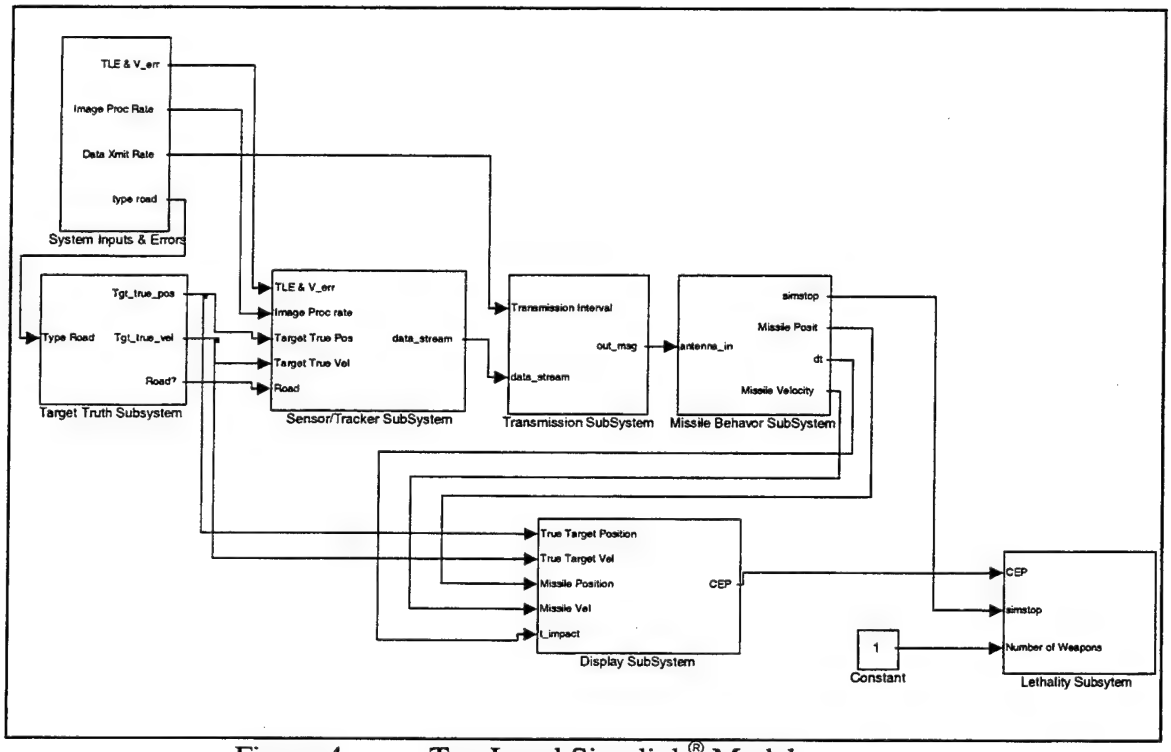


Figure 4. Top Level Simulink® Model

The output of the model is the radial miss distance of the weapon from the true target location at the predicted time of impact. Actual weapon kinematics and guidance have been simplified, so the resulting numbers do not reproduce the true Circular Error Probable (CEP)[20] for any particular weapon, but rather an approximation based on a generic weapon useful for relative assessments.

B. TARGET TRUTH SUBSYSTEM

The "Target Truth Subsystem" controls target movement during the simulation, and provides an output of true target position for analysis of accuracy. The target is a

[20] CEP is the most common measure of miss distance for calculation of weapon effectiveness. The CEP is a circle centered on the desired mean point of impact with a radius such that 50% of all weapons delivered lie within the circle.

single vehicle moving at a nominal 45 knots[21] in the 2-D ground plane. The target may move along a predetermined path ("road") or be allowed constrained random motion. Target motion is selected at the beginning of a simulation run, and remains constant for that simulation; a target may not initialize along a road then switch to random motion during a single simulation[22]. To speed up simulation 'runs', the target motion history for movement along a road was developed using a stand-alone MATLAB® Simulink® program, and stored in a MATLAB® compatible file[23]. At simulation initialization target velocity is extracted from the data file and integrated to produce target true position data.

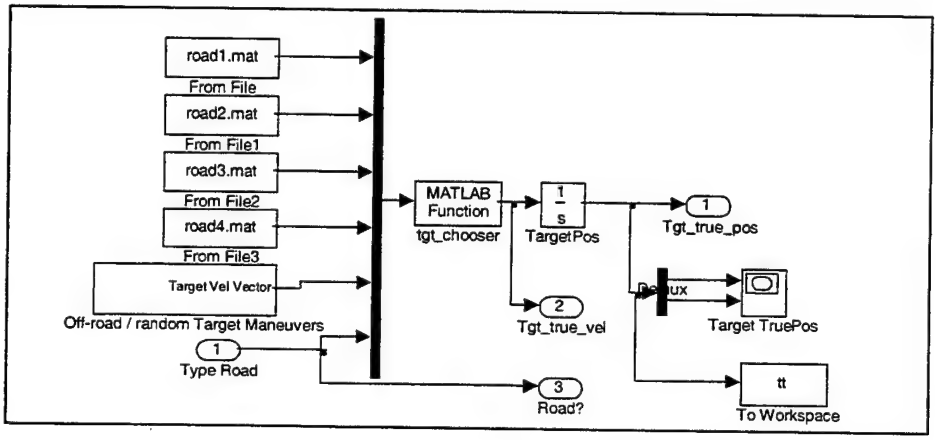


Figure 5. Target Truth Subsystem

1. Road Movement

The target may travel along either of two road configurations in the 2-D ground plane: (A) a straight road extending from the origin on a line toward the point (10,10), or (B) a road which begins at the origin and moves to a point (0,1) then a ninety degree turn to a point (0.5,1) then on an arc with a radius of 0.5 to a point (1,1.5) then along a straight

---

[21] 1 nautical mile is approximately 6076.10 feet , 1 knot, or nautical mile per hour, is approximately 1.1507 statute miles per hour (mph), 45 knots is approximately 51.8 mph or 83.3 kilometers per hour (kph).

[22] A basic assumption of the model is the target is unaware that a weapon has been delivered against it, so has no reason to initiate aggressive evasive maneuvering. This assumption is considered reasonable due to the standoff range and small signature of the weapons considered appropriate for this study.

[23] Target motion simulation files can be found in Appendix D.

line to the point (5,5) then along a straight line toward the point (10,10). Target velocity is scaled so that 1 unit of travel is equivalent to 1 NM.

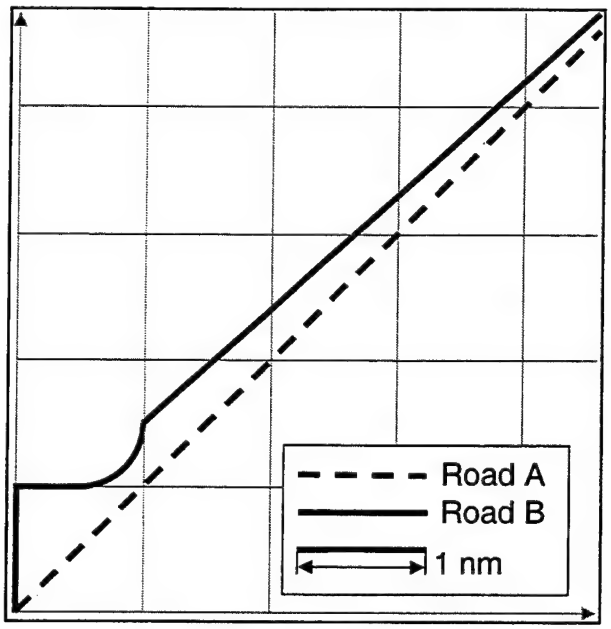


Figure 6. Road Map for Simulated Target Travel

On either road, target velocity may be constant or variable. At initialization the target velocity is set to 45 knots, or 0.75 nautical miles per minute. If a variable velocity profile is selected it is extracted from the appropriate data file and reflects a uniformly distributed random change of maximum +/- 3 knots per second to the velocity at each time step, resulting in the velocity profiles shown below. Note that for road profile 'B' the target slows down approaching the turn point at (0,1) and also at the turn point approaching the arc. The velocity profile is stored as part of the road data file[24] and is not adjustable during a simulation. Different velocity profiles may be generated and stored by using a different random number seed when creating the target velocity data file.

[24] The model uses data files, as shown in Figure 5, which correspond to the road profiles as follows: Road profile A, constant velocity, is stored in file "road1.dat", road profile A, variable velocity, is stored in file "road2.dat", road profile B, constant velocity, is stored in file "road3.dat", road profile B, variable velocity, is stored in file "road4.dat".

The combination of road and velocity profiles yield four different target motion files to choose from at the start of a simulation. Each file contains target motion data for 5 'minutes' of target movement. Target motion may be exactly replicated during subsequent runs since it is stored in the data file.

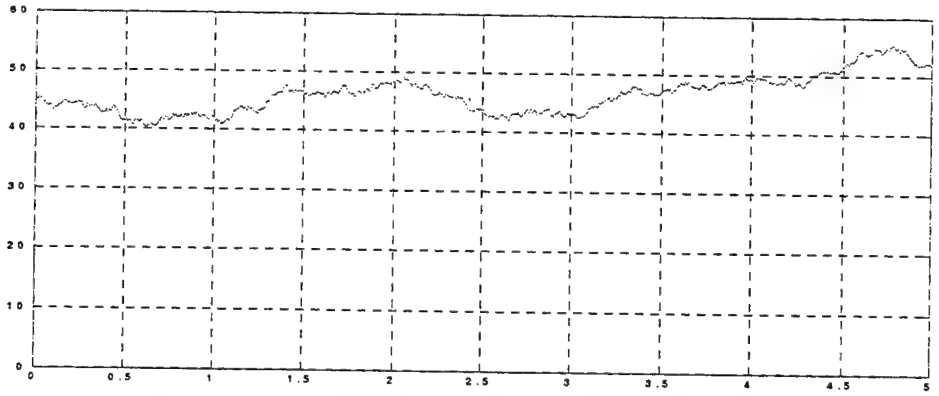


Figure 7. Variable Velocity Profile for Road A

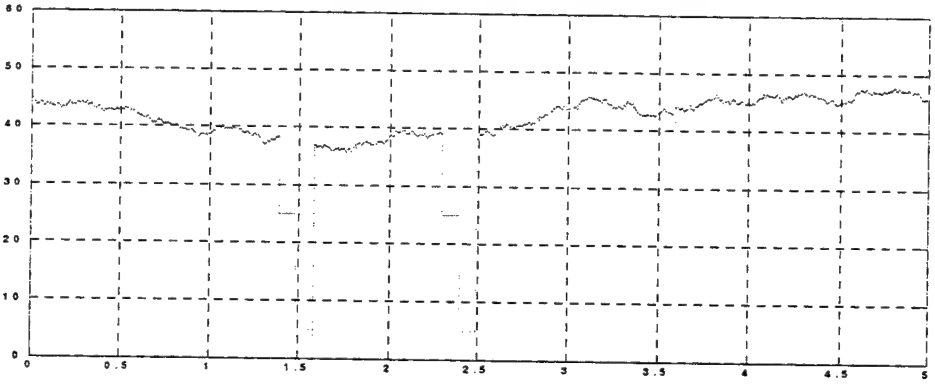


Figure 8. Variable Velocity Profile for Road B

2. Constrained Random Motion

This portion of the model was added for completeness of the research, but was not highly developed since the system has extremely limited utility when operating in this mode.

Constrained Random Motion is not stored in a data file, but is generated during each simulation. When not moving along a road, a target is allowed to move with limited freedom. Target motion rules were developed from the author's estimate of performance

of a typical automobile traveling at 45 knots. No definitive source for acceptable rules was found during initial research for this portion of the model, and as noted the development of detail in this module was limited due to expected low usage of this mode. This is not to imply that no rules exist; several modeling organizations list data files, but these were not deeply analyzed in this research.

The rules implemented to govern constrained random motion are briefly summarized here. Direction of target motion may be changed every 9.1 seconds. Using the pseudo random uniform number generator in MATLAB®, the target will make a turn of +/- 11.25 degrees 80% of the time. If a turn is made, it will be in the same direction as the last turn 70% of the time. Approximately 2% of the time that a turn is made, it will be a ninety-degree turn to represent evasive travel. Upon completion of a movement, the direction of the velocity vector will remain constant until the next turn opportunity. The velocity vector magnitude is initialized at 45 kts, and is adjusted using a uniform random change of maximum +/- 3 kts per second to the previous value at each time step as shown in Figure 9, but it is not modified to account for target velocity changes during a ninety-degree turn. A typical target motion plot is shown in Figure 10. Random number initialization data is not stored, so a run may not be replicated.

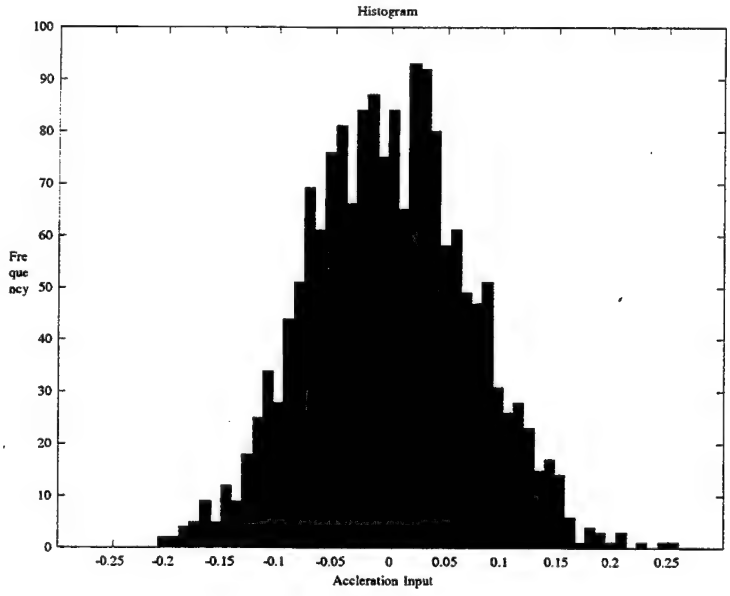


Figure 9. Histogram of Typical Velocity Change Inputs

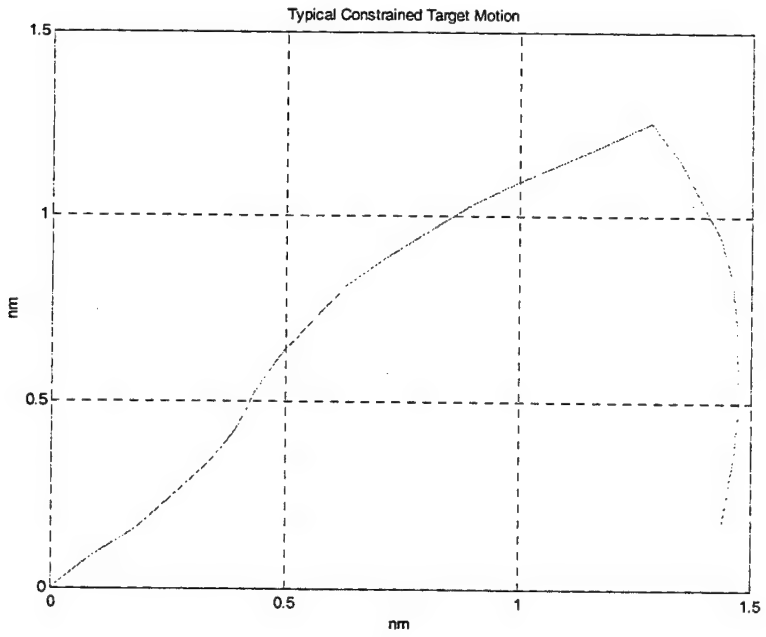


Figure 10. Typical 2-D Path for Target with Constrained Motion

C. SENSOR/TRACKER SUBSYSTEM

The "Sensor/Tracker Subsystem" interfaces with the target track via an assumed sensor, and provides predicted target motion data to the "Transmission Subsystem." This represents the DP/FS portion of the proposed system. The type of sensor used to deliver the 'image' to the system is not critical for the purposes of this simulation. The model assumes that an appropriate sensor, or suite of sensors covering several spectrums, provide a single or composite image which can be orthorectified, and registered, and from which target position and velocity can be extracted. The "Sensor/Tracker Subsystem" is further divided into an "Image Processor Subsystem", and a choice of either a "RoadMap" or "Target DR" subsystem. The choice of target prediction system is enabled by the choice of target motion in the "System Inputs and Errors" block, and is described below.

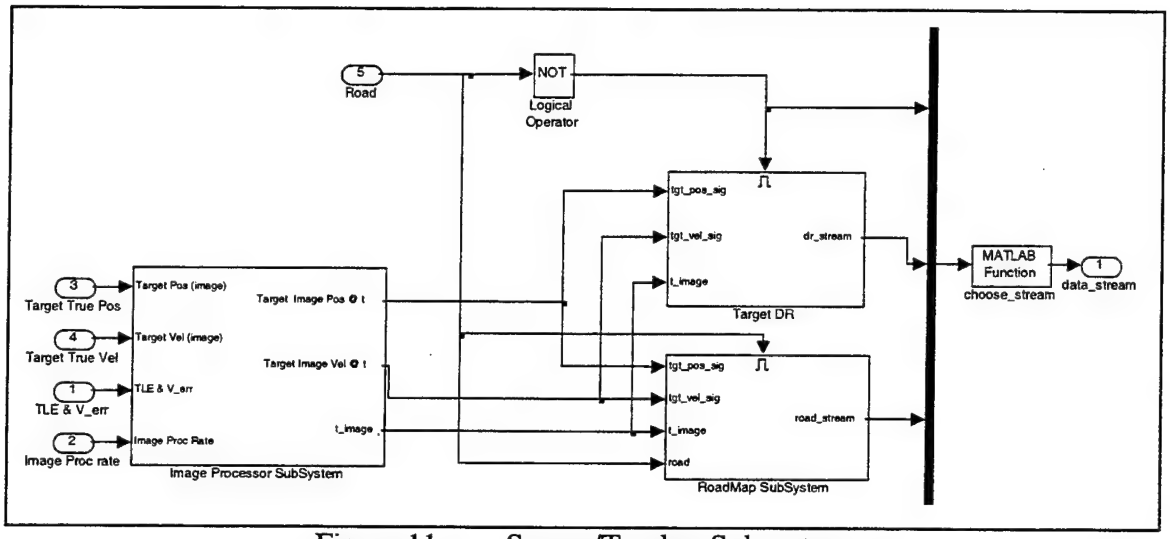


Figure 11. Sensor/Tracker Subsystem

1. Image Processor Subsystem

The "Image Processor Subsystem" represents the hardware and software that interface with the raw imagery received from the sensor. To represent this activity the target truth position and velocity are input and a uniform random number generator is used to add errors to the raw truth data[25] as shown in Figure 12.

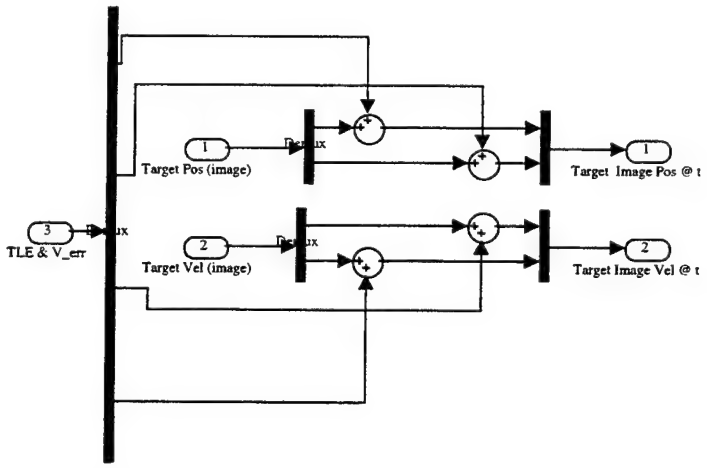


Figure 12. Adding Error to Target Truth Data

Figures 13 is a histogram of typical position error inputs for a single run, Figure 14 is a histogram of typical velocity error inputs.

---

[25] The error magnitudes are controlled by entries in the "System Inputs and Errors Subsystem" which will be described later.

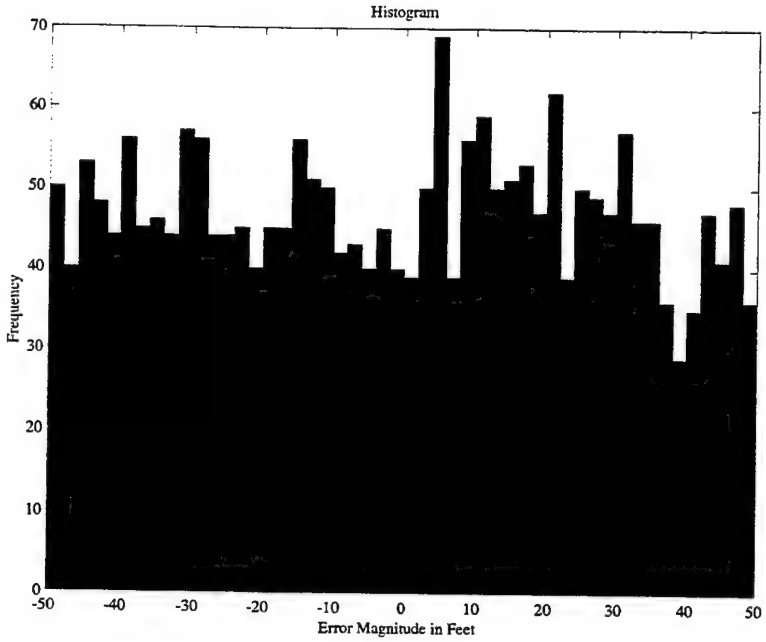


Figure 13. Typical Position Error Inputs for One Simulation Run

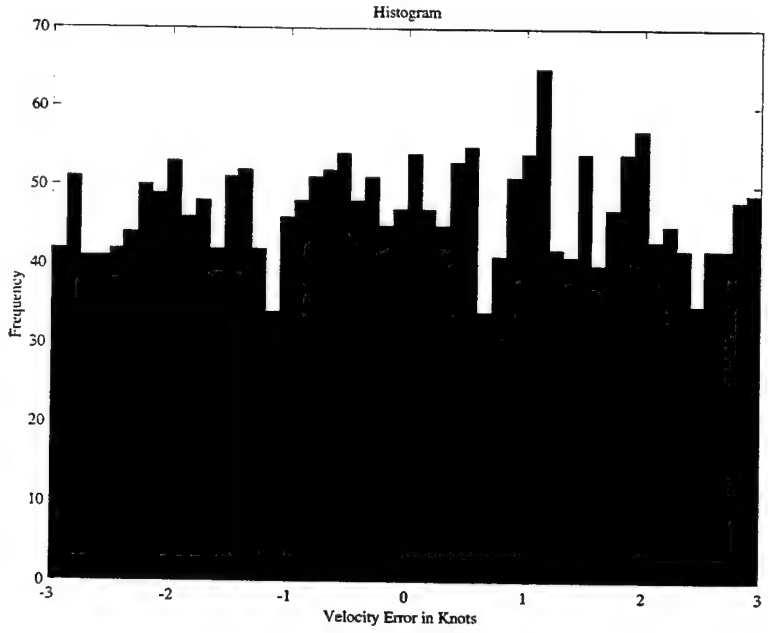


Figure 14. Typical Velocity Error Inputs for One Simulation Run

The "Image Processor Subsystem" also incorporates a delay module that holds the now-corrupted data for a nominal time before forwarding it for further processing. This

delay represents the period it would take to receive an image, identify the target, and extract target position data. The model assumes the system would build a track history file from which could be extracted a velocity, however this is artificially developed by corrupting the true target velocity data. The hold delay is not constant, but is constructed by taking a nominal fixed delay time input from the User, and adding a normalized random period before the next image is 'extracted', resulting in an aperiodic update rate, as shown in Figure 15.[26]

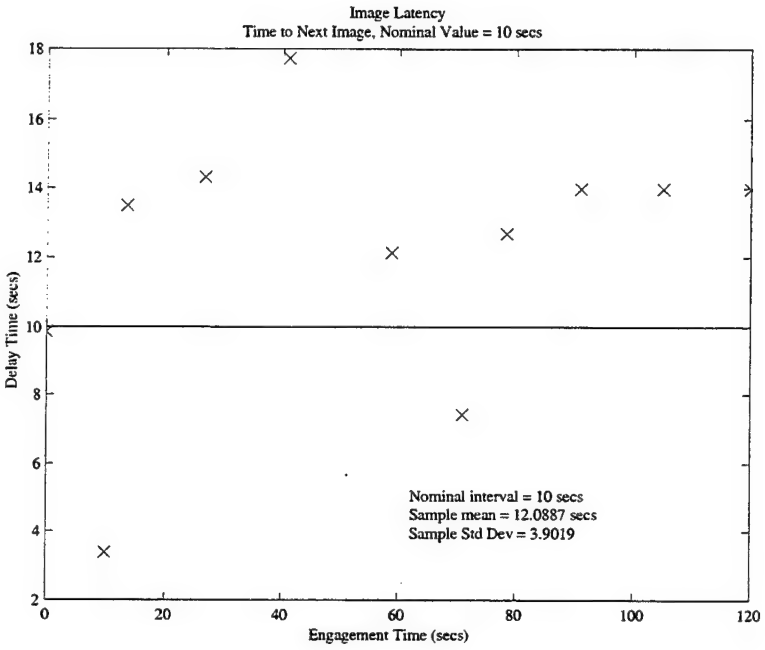


Figure 15. Example of Variable Image Latency

Image processing interval is a measure of the latency of the data. The interval is the time between when a new image is received, and when the data from that image is ready to be used. An interval of zero equates to no delay, i.e., the data is available instantaneously when the image is received. An interval of 10 seconds would mean when that data is available, it is based on an image that was received 10 seconds earlier.

[26] See Appendix C, file "latch_image.m" for implementation.

2. Road Map Subsystem

The "Road Map Subsystem" represents a smart system which receives the image "derived" data from the Image Processor, and uses a data base of geographic features to remove some error from the derived data, and predict future location of the target along a road network.[27] The road network model is an extreme simplification of the National Imagery and Mapping Agency (NIMA) vector product format (VPF) which would be used in the real world to provide topology and geospatial relationships of man-made features. The road models in the "Road Map Subsystem" are the same models used in the "Target Truth Subsystem" to develop target motion files, however the "Road Map Subsystem" does not have access to true target motion data (position or velocity).

Knowing the path of future movement allows some errors introduced during the image processing to be reduced. Since the target is known to be traveling on the road, and the vector model approximation to the road is available to the system, position error is minimized by ensuring the target trajectory stays 'on the road' during predicted movement. The target velocity provided to the system also contains errors. Using the knowledge that the target is moving on a road, however, allows the model to assume the true velocity vector direction must be aligned with the road axis. Therefore, the provided vector direction is discarded and the magnitude is aligned with the road axis vector to provide a new velocity.

Calculation of the future position of the target begins with the extraction of the target position in the "Image Processor Subsystem". This estimated position is compared to the approximation of the road from the database, as shown in Figure 16.

---

[27] "Road" as used in this model means any predictable path of travel which can be represented by a series of vectors either stored in a developed database, such as the NIMA VPF, or prepared on the spot by the system operator based on previous wheel tracks, obstacles, or other information available in the image, and from which prediction of expected target motion can be made.

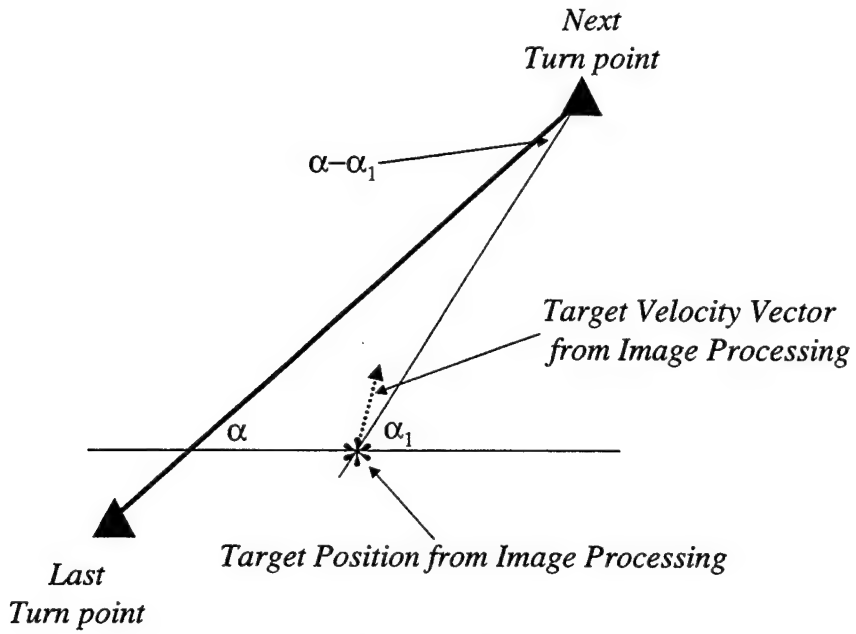


Figure 16. Estimated Target Position

The model[28] calculates the angle and the distance from the estimated position to the next turn point, and the angle and length of the nearest road segment. Using the magnitude of the velocity vector to determine how far the target will move in the first time step, the next position in the x-direction is calculated as:

$$\cos(\alpha) * \cos(\alpha - \alpha_1) * (dist\_to\_turn - velocity\_magnitude_{1\_time\_step}) \qquad (3.1)$$

and in the y-direction as:

$$\sin(\alpha) * \cos(\alpha - \alpha_1) * (dist\_to\_turn - velocity\_magnitude_{1\_time\_step}) \qquad (3.2)$$

which places the predicted target location after one time step on the road, with the velocity vector aligned with the road, as shown in Figure 17.

[28] See Appendix C, file "getmap.m" for implementation.

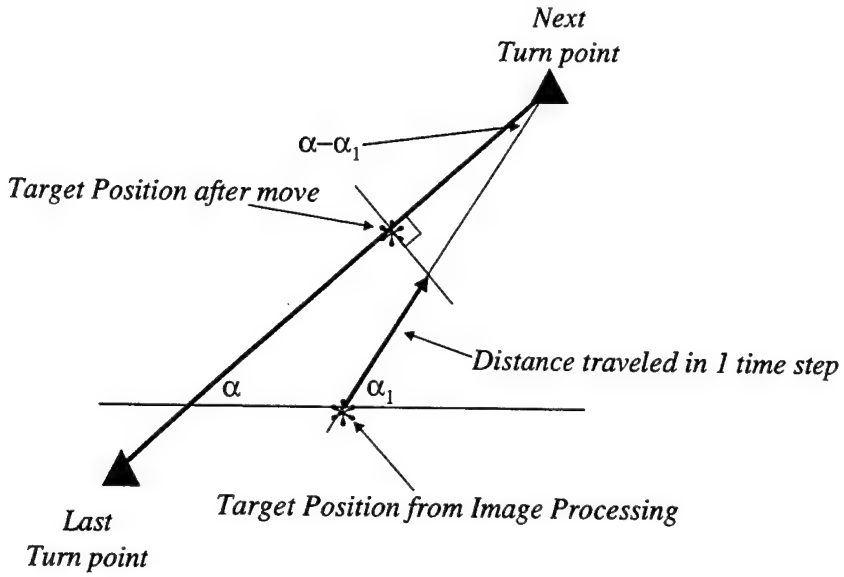


Figure 17. Predicted Target Position after 1 Time Step

The problem is slightly more complex when the initial estimated position is just prior to a turn point[29], and the distance to be moved will go beyond the turn point. As shown in Figure 18, the model first calculates distance to the next turn, then subtracts this from the distance to be traveled during the move. Any excess distance is applied along the axis of the next road segment. If a move 'steps over' more than one road segment, which may happen when the vehicle is moving along a curved section of roadway (as represented by the arc in road profile B), the length of each road vector segment is sequentially subtracted from the total distance to be moved, until the excess travel distance remaining is less than the length of the next road segment. The excess travel remaining is then applied as in the case without passing a turn point.

Once the first predicted target position is placed on the road, subsequent estimates of target position are calculated by assuming a constant target velocity, and advancing each position in the direction of the current road segment, using the same technique as above.

[29] A "turn point" is the same as the end of a vector segment.

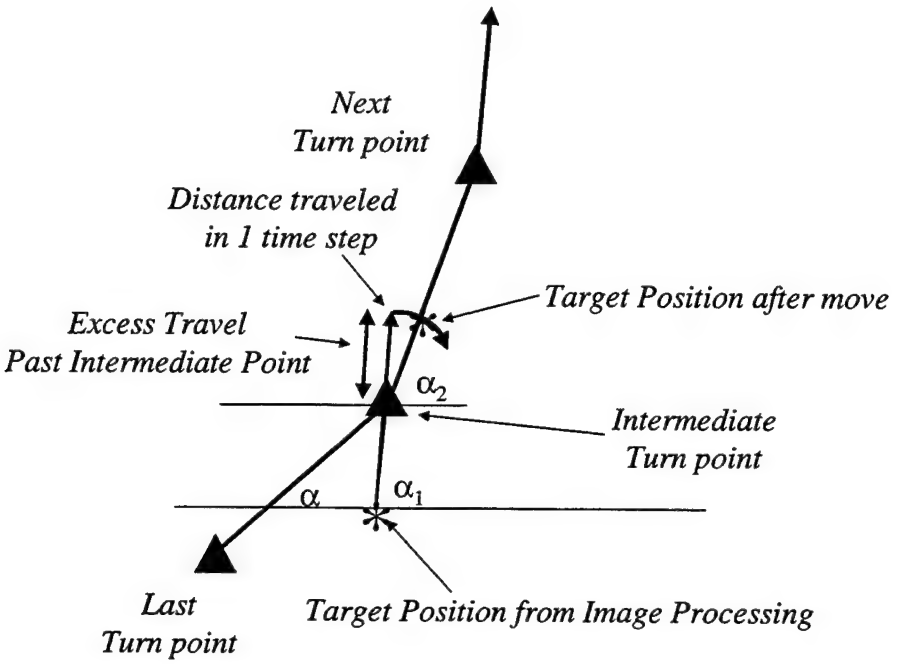


Figure 18. Predicting Target Position Beyond One Turn Point

3. Movement Prediction

Generation of future target position predictions is done in a ground station with access to large databases of geographic and cartographic data, and links to multi-sensor sources. The information provided to the weapon is the output of this information fusion, as noted in Chapter II.[30] Ideally, the system would provide predicted target position from the time of the most current image to the predicted time of weapon impact, including some margin for time-of-flight uncertainty. Each predicted point should be spaced close enough to its predecessor and successor to minimize interpolation errors, and should have an associated target velocity vector to support that interpolation of target position between contiguous data points.

However, as noted in Chapter II, the weapon does not transmit its own position to any other part of the system, so it is not possible to accurately know the time of predicted impact. While it may be possible to use the estimate of the duration of flight at the time of weapon launch, this requires a potentially significant amount of coordination and data

---

[30] Use of a ground station as the fusion center should allow easier system upgrade/update, and access to larger databases, than would be available in a processor onboard a weapon.

exchange. Another limiting concern is that for most engagements it may be necessary to send multiple weapons following different routes[31] to the target. Finally, it would be desirable to provide data to weapons enroute to multiple targets, each of which would need its' own data stream. If possible, weapons proceeding to the same target should be able to use the same data stream information. These issues lead to the consideration of a non-optimized data stream length to allow enough accuracy but without requiring the entire available link bandwidth.

For the purposes of this research it was decided the system should be able to provide data on ten separate targets. A unique track identifier will identify individual targets, and this identifier will be loaded prior to release into each weapon assigned to that target to allow the weapon to identify the appropriate data message coming over the link. Multiple weapons going to the same target can receive the same data stream of predicted positions, and calculate their own intercepts.[32]

The next consideration was the amount of data required for the predictions. A target position and velocity must be defined at each in two dimensions[33] with good accuracy. The research assumed that a one-axis position would require 32 bits to yield the necessary accuracy, and one-axis velocity vector would require 16 bits. For each predicted point two components each of position and velocity are required. Additionally a time using 8 bits would be used to identify the data, so each point would need approximately 104 bits of information. Since the research considered multiple targets as a core capability of the system, this amount of data was reviewed to determine if it could be reduced. The less data required per predicted point, the more points can be provided in a given time slot on the data link. To decrease the amount of data needed at each point

---

[31] Either by programming a different route into the weapon or by launching weapons from non-collocated aircraft. This would complicate the target's self defense problem, and potentially improve the probability of mission success.

[32] The model currently supports only one weapon, however the timing issues involved in attacking of ten targets are considered to determine the effect on one weapon as representative of many weapons vs. many targets.

[33] The model is limited to a 2-D engagement. The actual system would need to provide data in 3-D.

the model eliminated the need to transmit velocity data by spacing the position points equally in time. Thus velocity data can be calculated by comparing the distance between two sequential data points and dividing that by the fixed time interval[34]. The information needed for one data point was reduced to 72 bits, a 31% saving.

To determine the number and spacing of predicted points necessary to assure reasonable accuracy, the scale of the target movement must be understood. At 45 kts, the target will move 75.9 ft each second, or 4556 ft in one minute. The weapon, traveling at 420 kts, will cover 708.7 ft each second, or 42,525 feet in one minute. Assuming the image can take up to 30 seconds to process (image latency), the data stream must be at least 30 seconds long just to predict where the target is when the weapon receives the information. 30 seconds was chosen as the minimum acceptable length, at which time the weapon is still over 21,000 feet away, with sufficient time to maneuver and execute a terminal engagement.

At the maximum end the stream could conceivably contain position prediction estimates for several minutes into the future. There are two limitations that helped define the upper bound for this model. First, the model assumes the target maintains a constant velocity through the period of the data stream, which is a known error source. Conservatively assuming there is a one-knot constant error, in one minute the estimated position will be up to 101 feet[35] from the true position. The relative effects of a one-knot constant error are shown in Figure 19.

From Figure 19 it can also be seen that, assuming a normal distribution of velocity errors over many trials, the variance of the location error will increase the farther

---

[34] This solution was found by considering the principles of TRIZ, a Russian acronym for a phrase that translates as the "Theory of Inventive Problem Solving". In this case, Principle #40 *Matrioshka* (Nesting) was used to put the velocity information into the stream data implicitly rather than explicitly. This technique would work equally well for a non-constant target velocity, as long as the points were equally spaced in time.

[35] The shape of the area constrained by the error boundary would be roughly elliptical if it the target was not constrained to move along a linear path on the road.

into the future the time for which the prediction is made[36]. Obviously, providing more data does not necessarily equate to providing more accurate data. For the model the maximum stream length was set at 60 seconds of data, since at this point the uncertainty radius for a one-knot error is larger than the damage mechanism of most weapons considered for this study.

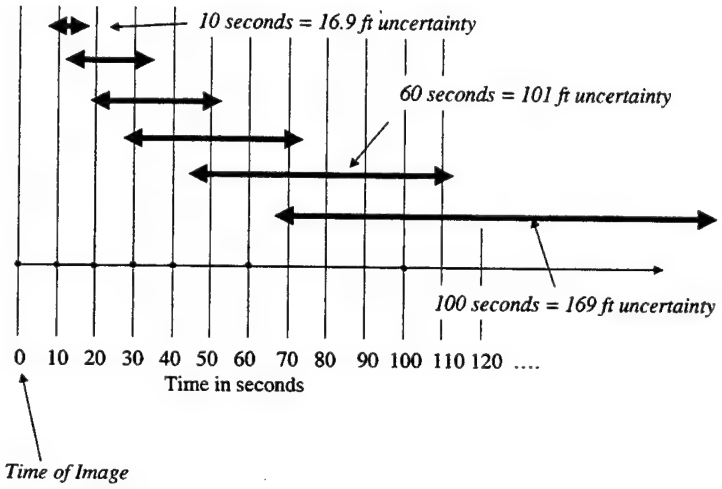


Figure 19. Effect of 1-Knot Constant Error

Sixty seconds will provide at least 30 seconds of useful data, at which point the weapon is still over 3 NM away from the target with plenty of time to make final maneuvers. With less image processing latency, this range increases to a maximum of 7 NM (if latency is zero in instantaneous image processing).

Limiting the data prediction to 60 seconds after the image is obtained results in the introduction of a known error source into the missile intercept estimation when the missile is still beyond the last predicted point. The magnitude of this temporary error is dependent on the road geometry. For the worst case, if the target is still beyond the 60 second stream of data, and the image derived data update rate is once every thirty seconds, the maximum cross track deviation (from an instantaneous ninety-degree target

[36] This characteristic of forecasting is apparent in any situation, for example in meteorology, or the stock market.

heading change occurring immediately after the last update is received and made at constant velocity) would be approximately 0.53 NM between the predicted position and the next updated position, as shown in Figure 20. The capability to maneuver to the target should be well within the guidance and control capability of any weapons considered during this study.

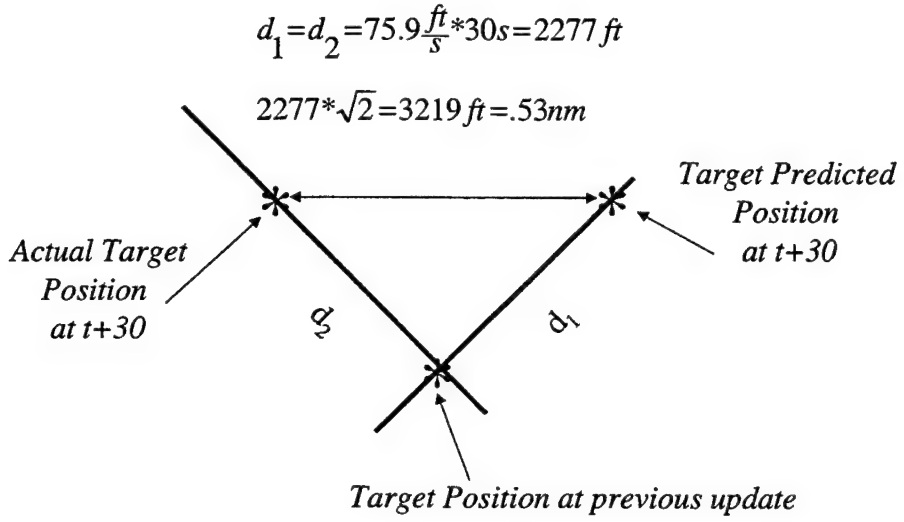


Figure 20. Difference Between Predicted Positions

Since most land vehicles would slow down approaching the intersection to safely make the turn, d_2 would be less than d_1, and the actual cross track separation between updates would be less, resulting in minimal impact of the predicted impact location on weapon flight path management. In any case, once the missile computes an impact time inside of the 60-second data stream, the weapon has more accurate movement predictions available, and can make final compensation. Similarly, effects of velocity changes on predicted impact point would be minimal outside of the 60-second data stream, and very small within.

4. Data Message Format

The output of the subsystem represents target location at the time of the image, and includes 60 data points spaced one second apart to predict future target motion. For the last data point, the target velocity will be supplied to enable the missile to compute its' own dead reckoning point of intercept. The number of necessary predictive data

points was determined by considering the maximum desired length of the data message to the weapon, but the interval between data points was determined by considering the implementation of the formatted message. A data interval of one second between points was deemed acceptable since the target position changes between data points would be limited[37]. The real-world system proposes compatibility with a Link-16 Tactical Data Link to be used in conjunction with the Multi-functional Information Distribution System (MIDS) on the F/A-18. While several transmission formats are available using Link-16, Variable Message Format (VMF) may prove most practical and provides a throughput of approximately 238 KBPS[38].

The modeled weapon data stream is structured as follows:

Word contents	bits
time of image (t)	8
position of target (x-dir. or Lat.) (t+60)	32
position of target (y-dir. or Long) (t+60)	32
velocity of target (x-dir. or Lat.) (t+60)	16
velocity of target (y-dir. or Long) (t+60)	16
words 6 – 190 time of prediction	8 (480)
predicted position (x-dir. or Lat.)	32 (1920)
predicted position (y-dir. or Long)	32 (1920)
Total	**4424 bits**

The initial word in the message would be an address or track number to trigger the weapons assigned to that target to accept the data. This address is assumed to be 16 bits, for a total of 4440 bits, or approximately 1.865e-2 seconds of data stream.

---

[37] A data interval of 0.5 seconds was also considered, but time precluded consideration during this research.

[38] Unclassified overview of Link-16 hardware and data was obtained from the United States Naval Academy Division of Professional Development webpage located at http://prodevweb.prodev.usna.edu/Seanav/. Link-16 information found on this webpage also appears on pages at the SPAWAR site.

5. Target DR Subsystem

The "Target DR Subsystem" is used when the target is not moving along a predictable track or road; the target is free to move in any direction. Due to the expected low utilization of this type of free-motion engagement, the implementation of this part model is extremely simplified, and the model does not attempt to predict future motion. The model assumes the target will remain at the same velocity and direction computed from the last image.

D. TRANSMISSION SUBSYSTEM

The "Transmission Subsystem" represents the transmission, and relay nodes that receive the data stream from the "Sensor/Tracker Subsystem" and periodically transmit the data to the weapon. For this model it is assumed the transmission will utilize Link-16 in the real world, although explicit modeling of Link-16 characteristics was not attempted.

The "Transmission Subsystem" stores the most current data received until it is time for the next transmission. If multiple updates occur between transmissions, only the most current data is transmitted. Transmission is modeled as perfect/instantaneous except that a random dropout rate of 2% is included, which represents the combined effects of attenuation, positioning, and interference that could occur within a mission.

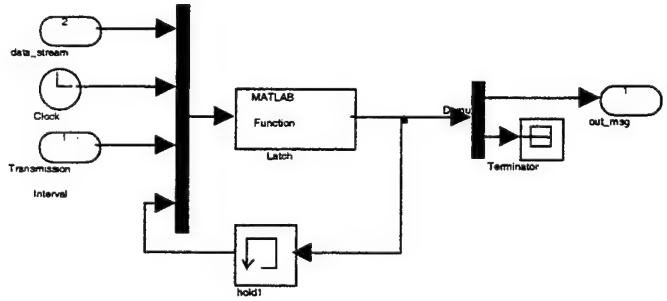


Figure 21. Transmission Subsystem

Data transmission interval is a periodic spacing that can induce a latency of up to the interval specified. As in image processing, an interval of zero would equate to no delay; the data is transmitted instantaneously. However, an interval of 10 seconds means only that the next data transmission happens 10 seconds later. The data contained in that

transmission could have been buffered anywhere within that 10-second period, so may induce an additional 10 second latency, or less. An example is shown in Figure 22.

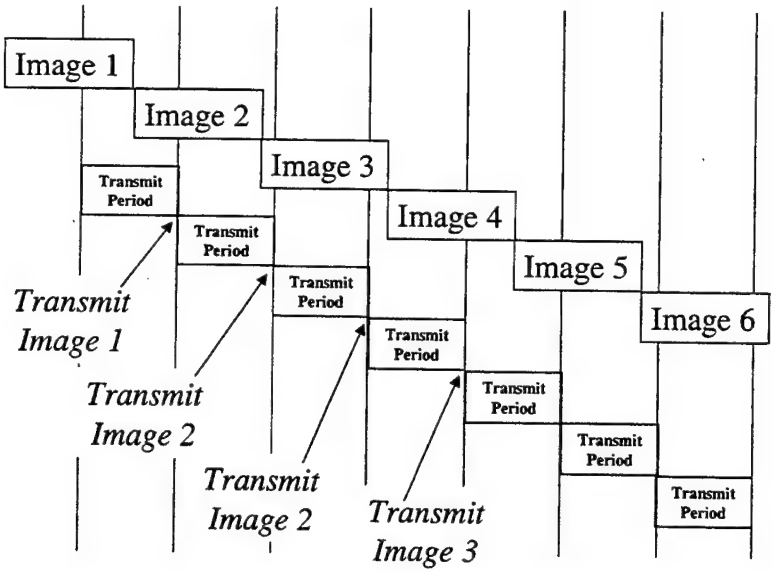


Figure 22. Example of Cascading Latency Effects

'Image 1' is received and processed. The extracted data is ready for transmission, but the transmit time is in the future, so the data is buffered, and an additional delay is added to the data. The same effect is seen in 'Image 2', although the transmission hold is shorter. However, at the next transmission time, 'Image 3' is not ready so the transmission buffer holds only data from 'Image 2', which is resent. To minimize the cascading effect, the transmission interval should be much shorter than the image interval.

E. MISSILE BEHAVIOR SUBSYSTEM

The "Missile Behavior Subsystem" represents an idealized weapon assigned to attack a single target. Kinematics of a generic weapon are represented by computing a lead navigation solution[39] and tracking the convergence of the computed position along that navigation path and the predicted location of the target from the "Sensor/Tracker Subsystem." The weapon does not transmit its own position or predicted intercept information to any other part of the system.

---

[39] The algorithm used was adapted from Zarchan, *Tactical and Strategic Missile Guidance, Second Edition*, chapter 2, listing 2.1, and modified for the MATLAB Simulink environment.

Proportional, or lead computing, navigation is desirable in this type of system since it reduces time of flight of the weapon and energy use compared to a constant line of sight, pure-pursuit or beam-rider guidance. Essentially, the weapon computes a future position of the target and alters course to intercept based on iterated time-to-impact. When the future position is outside of the 60 seconds of predicted positions provided in the data stream, the missile assumes the target maintains a constant heading and velocity from the last predicted position, and bases the impact point on that information.

The model does not attempt to align weapon heading with target ground track to affect probability of kill. In actual implementation weapon effectiveness may be improved if the attack axis is oriented to minimize weapon miss distance using statistical data on the individual weapon's probable errors in range and deflection.[40] Aligning the attack axis will, however, increase the time of flight of the weapon, and increase the use of weapon energy, so any decision to do so must include a trade-off study between these factors, which was considered outside the scope of this research. Additionally, this model uses a 'flat earth' representation of the battlefield, and does not include any terrain or other features that may require shaping of the attack path to ensure the weapon reaches the target area. Since each weapon would approach this problem differently, modeling was considered outside the scope of this research. Neither of these simplifications is considered detrimental to the relative measure of effectiveness derived from the overall model.

As noted earlier, the model sets an initialization point for the weapon at a random point at least 5 NM but not more than approximately 35.4 NM away from the origin. There are also two random errors in the missile model. The first error is the initial heading from the launch point to the target generated by MATLAB® from a Uniform distribution with a maximum deviation of +/- 10 degrees, which is quickly removed by the lead computing algorithm, but may impact accuracy during short range launches. The

---

[40] Range Error Probable (REP) and Deflection Error Probable (DEP) are the distances from the desired mean point of impact (DMPI) to one of a pair of lines perpendicular to the respective direction, and equidistant from the DMPI, spaced such that 50% of all impacts are between the lines. REP and DEP are unique to each weapon.

second error is a 2-D random acceleration input that represents wind disturbance of the weapon velocity, which is initialized to 420 knots, and uniform random wind accelerations are applied to deviate from this velocity. This error source manifests itself in varying time-to-target estimations, which are used to calculate the impact point. Finally, the weapon model lacks kinematic control surface responses, and the self-navigation errors, although these are assumed to be smaller than the target location errors for the types of weapons assumed in this analysis.

The "Missile Behavior Subsystem" initiates the 'stop simulation' sequence when either of the following conditions is met:

1. The model calculates the separation between the weapon's own position and the estimated position of the target from the information in the data stream and the "Target DR" module in the "Missile Behavior Subsystem". The simulation will stop when the calculated separation is less than 0.1 nautical miles[41], and the receiver has not received a data stream update in the last 0.002 'minutes', and the separation at the current time step 'n' is more than the separation at the last time step 'n-1', indicating that the weapon has passed the minimum radial distance from the target.[42] Since this closest point of approach is based on the estimated target position, the actual miss distance is calculated by the "Display Subsystem" based on target position from the "Target Truth" subsystem.

---

[41] 0.1 nautical miles was chosen to ensure that 'false positive' hits would be minimized. There is a case when changes in the target relative position during data stream updates, notably early in the simulation when the weapon has not completed maneuvering toward the initial predicted impact area, that the separation at the 'n+1' time step is increasing, but it is due to relative position of the weapon and target which will wash out quickly. By the 0.1 nautical mile circle, the weapon will be approximately 0.77 to 0.96 seconds from impact, well within the '60 second' data stream and will have much more accurate position data which will reduce this probability. In a real weapon the data update would likely have been 'locked out' at this point due to weapon control system and time-constant effect of the response capability.

[42] Note that an actual weapon would have flown to a calculated impact point, and would not 'fly by' the target.

2. If the weapon-to-estimated target separation does not enter the 0.1 nautical mile circle around the estimated target position the run is a 'miss'. The 'stop sequence' will be initiated at 4.9 'minutes' to ensure the simulation does not read past the end of the target motion data file. This case is most likely to occur if the weapon is navigating to an impact point and receives an update such that the target position is significantly changed due to a heading or velocity change, and the missile cannot maneuver to reach the target.[43] In this case the miss distance will be unusually high and must be accounted for as such in the data.

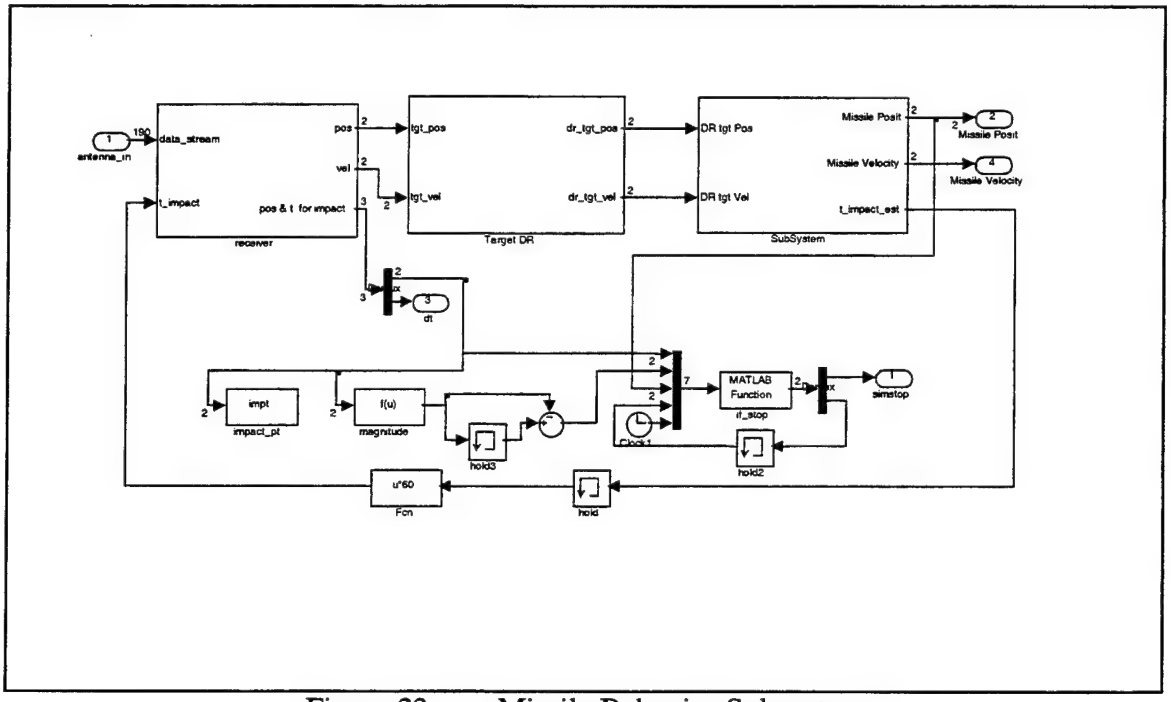


Figure 23. Missile Behavior Subsystem

[43] The weapon is restricted to approximately 10g's during lateral maneuvers.

F. INPUT AND ANALYSIS SUBSYSTEMS

There are three subsystems that are used as input and output controllers:[44]

1. System Input and Errors Subsystem

This subsystem block was used in lieu of a graphical user interface to adjust the primary simulation variables during the simulations. The variables include target location error, target velocity error, image processing interval, data transmission interval, and selection of the road data file for target movement during simulations.

Target Location Error is a magnitude selected from a uniform distribution of numbers within the +/- limits of the input value. To increase the randomness of the net input, a separately generated TLE is applied to the image position in each axis individually, therefore the net TLE is the represented by:

$$TLE_{Total} = \sqrt{TLE_{x-axis}^2 + TLE_{y-axis}^2} \qquad (3.4)$$

with the result the TLE_{Total} measured as a magnitude from the true location will be greater than the individual input value in either axis, unless one axis is exactly zero. Target Velocity Error is input and used in a similar manner.

Image processing interval is a measure of the latency of the data. The interval is the time between when a new image is received, and when the data from that image is ready to be used. An interval of zero would equate to no delay; the data is available instantaneously. An interval of 10 seconds would mean when that data is available, it is based on an image that was received 10 seconds earlier. Data transmission interval is a periodic rate at which the data buffer will be released to the weapon receiver.

Road data file is the choice of the road profile to be used during a particular simulation.

2. Display Subsystem

The "Display Subsystem" accepts position and velocity data from the "Target Truth" and "Missile Behavior" subsystems, and calculates the radial miss distance (RMD) from the missile location at time of impact to the true target position. The miss

---

[44] See Appendix C for model diagrams.

distance is calculated by tracking the 'nth' and 'n-1th' position[45] of the true target and the missile as shown in Figure 24.

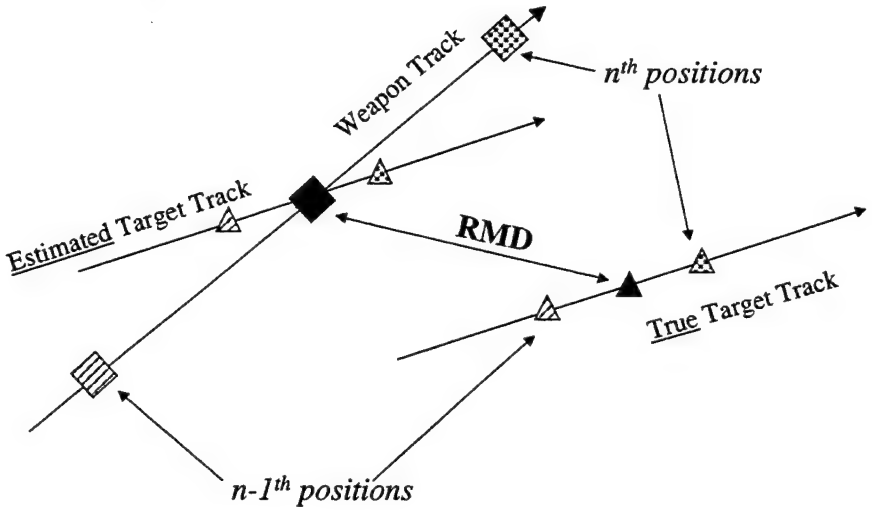


Figure 24. Calculating Radial Miss Distance

The "Display Subsystem" uses the estimated target velocity to calculate the intra-step time when the weapon will cross the estimated target track.[46] This delta time is used to advance the weapon and true target positions from their respective n-1 positions, and the true radial miss distance is calculated by taking the root sum square of the difference in these positions.[47]

3. Lethality Subsystem

This subsystem receives the RMD from the "Display Subsystem" and calculates an estimated single sortie probability of damage (SSPD)[48] using generic target dimensions and warhead measures of effectiveness. SSPD is calculated using an adaptation of Joint Munition Effectiveness Manual (JMEM) methodology for unguided

---

[45] The simulation is stopped on the 'n' iteration when separation increases, so the last point before impact is the 'n-1' positions.

[46] In a real engagement this would be the intended impact point.

[47] See Appendix C, file "true_cep.m" for implementation.

[48] SSPD is the probability that a single weapon will inflict a desired level of damage on a target based on the damage function/effectiveness indices.

weapons[49] and is displayed in the MATLAB® command window with RMD at the end of the simulation.

The "Lethality Subsystem" was included for completeness of the model, but the use of generic (Unclassified) data for target and weapon information limits its utility, and results of the simulation are not considered valid indications of system effectiveness.

This subsystem also issues the "stop simulation" command.

---

[49] Naval Postgraduate School Report, *Report on the Applicability of Current JMEM Delivery Accuracy (DA) Methodology to JDAM*, by M. Driels shows that JMEM Guided Weapon methodology may not be appropriate for GPS weapons without seekers.

IV. SIMULATION RESULTS

A. OVERVIEW

The final version of the model was completed on about 26 January 2001. To enable multiple data collection runs, a copy of the final model was made, and a separate MATLAB® file was developed that called the model and varied input parameters for each test case run.[50] Unless otherwise noted one 'run' consisted of 100 repetitions of the model sequentially on road profile 'A', road profile 'A' with variable target velocity, road profile 'B', and road profile 'B' with variable velocity, yielding 400 data points per 'run'[51]. Tabulated results can be found in Appendix E.

1. Movement Prediction

Target motion was predicted based on data extracted from an 'image'. The extracted data was known to contain errors in both absolute location, and velocity. As noted in Chapter III, the algorithm attempted to minimize these using a vector representation of the road the target was traveling. The output of the algorithm was a stream of 60 data points, and a final velocity vector correlated to the 60^{th} data point. Figure 25 shows an example of the predicted locations for a single typical run of the model. Note the first data point - the value actually extracted from the imagery module lies 'south' of the road, and the algorithm adjusted the remainder of the points to correspond to the arc of the turn, presumably reducing predicted location errors.

2. Example of Intercept Geometry

During each engagement the weapon computed a lead heading to intercept the target based on relative position and velocity estimates derived from information

---

[50] The model was run on a Micron Pentium III with a processor speed of 500 MHz running the Windows NT version 4.00.1381 operating system registered to the Naval Postgraduate School.

[51] The model was not coded optimally, therefore one typical 'run' required approximately 6 hours to complete.

contained in the received data stream, and the weapon's calculated time of impact. Figure 26 shows a typical engagement scenario on road profile 'B' with variable target

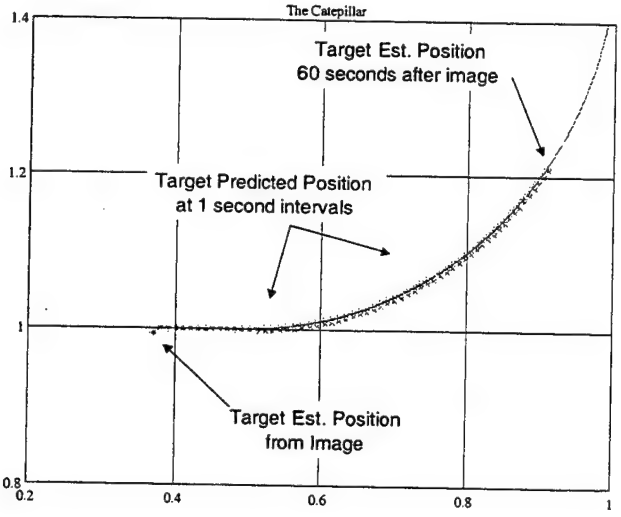


Figure 25. Predicted Target Motion Points

velocity. When the engagement was initiated the weapon navigation algorithm used the last of the predicted target positions and the associated velocity data to compute an intercept point. It then created a pseudo-position for a ghost target which was located back from the intercept point a distance equal to the estimated velocity multiplied by the time to impact so the ghost target straight line movement was tangent to the predicted intercept point. As the engagement progressed the last predicted data point in the 60 second data stream moved on to the arc, and the ghost target reflected the change, as seen in the track change. Finally the predicted intercept moved off the arc, and the ghost target track aligned with the final road segment.

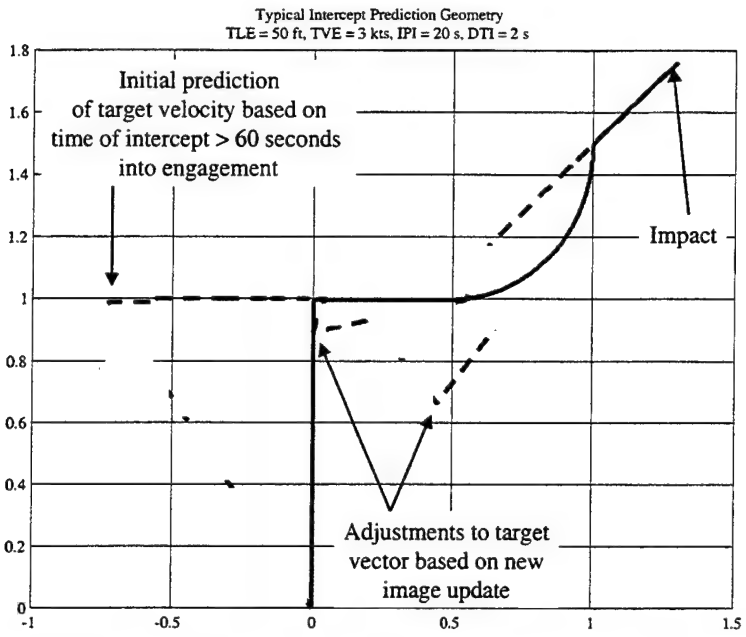


Figure 26. Missile Prediction of Intercept, Road B with variable Velocity

3. Predicted Impact Points

Throughout the engagement the weapon refined the predicted impact point at each successive iteration. Figure 27 shows the same engagement as above, with predicted impact points highlighted by asterisks.

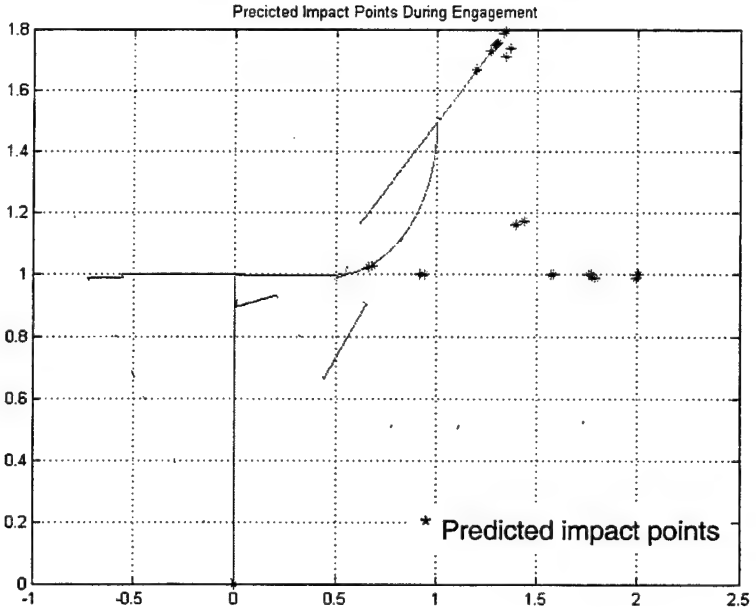


Figure 27. Predicted Impact Points

It is apparent the impact points converged as the time-to-impact estimate moved within the 60-second data stream. Figure 28 shows an expanded view of the final positions for this engagement about the time of predicted impact. There is still some 'jitter' in the position due to the uncertainty in the target velocity, and the effect of 'wind' from within the missile behavior model. Note: the target and weapon positions extend beyond the calculated impact point because the simulations continued by design to allow miss distance calculation to occur.

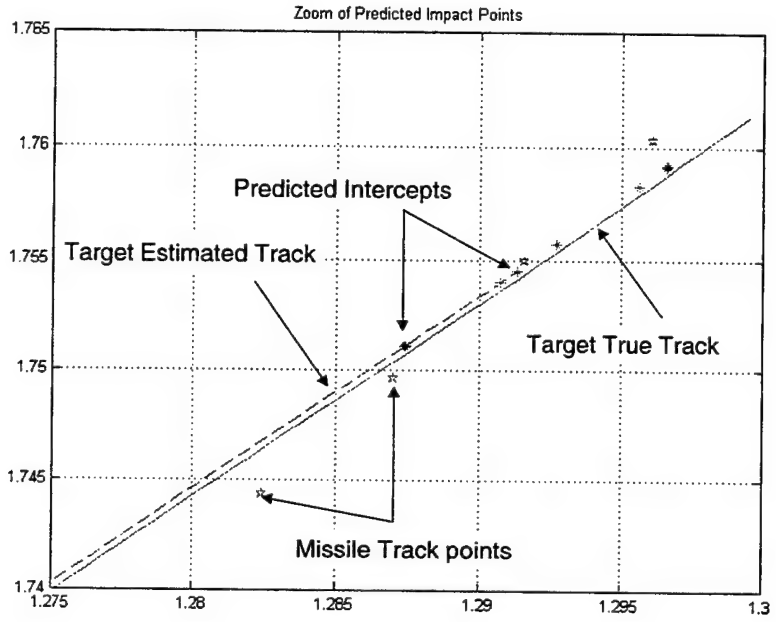


Figure 28. Expanded View of Predicted Impact Points

4. Calculation of Radial Miss Distance

The model uses the difference between the actual target location at impact, and the position of the missile at impact to calculate the radial miss distance. As discussed in Chapter III, the model interpolates position data between the points based on time of predicted impact. The radial miss distance for this engagement is depicted in Figure 29.

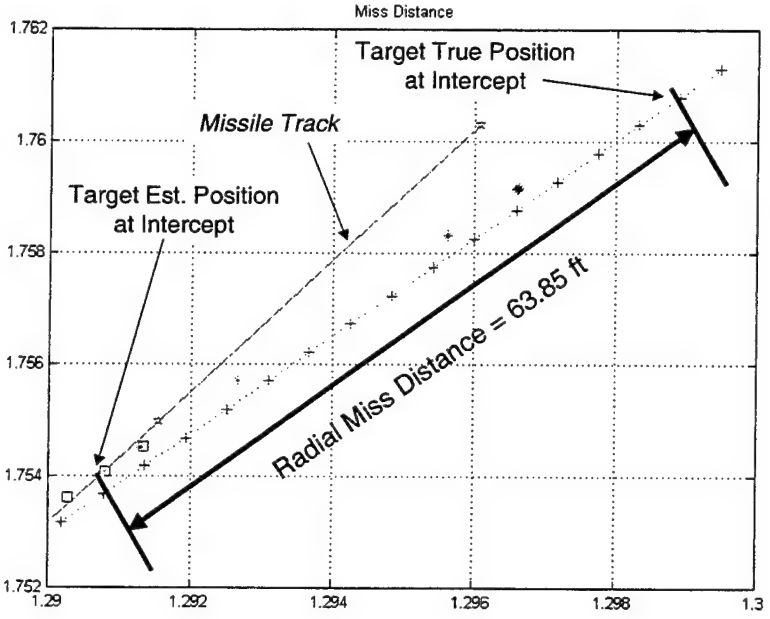


Figure 29. Example of Radial Miss Distance

B. DATA ACQUISITION PLAN

The purpose of the model was to assess requirements for GPS-aided low cost conventional standoff weapon to be used to attack time sensitive moving targets without the aid of a weapon-based terminal seeker. The variables considered to be critical parameters during this phase were Target Location Error (TLE), Target Velocity Error (TVE), Road Profile, Image Processing Interval (IPI), and Data Transmission Interval (DTI), as shown in Table 1.[52] The basic methodology was to use a Monte Carlo[53] technique to change one variable at a time, run a series of model test cases, and store the

---

[52] This study was conducted with unclassified data assumptions. The values chosen for analysis do not represent or imply any capability of any particular sensor, database, weapon, or technique. They were chosen solely by the author of this paper based on input values that would produce results within acceptable levels.

[53] 'Monte Carlo' was a term coined on the Manhattan Project to describe probabilistic methods applied to determine outcomes of random combinations of events. The method uses a series of 'chances' each of which can be defined by a probabilistic input variable, or variables, and uses the outcome of many 'chances' to characterize the likely response of the output, which has an initially unknown probability function.

output. Only limited in-process data analysis was made to ensure the CEP results appeared 'reasonable' before initiating the next set of data runs. This allowed for a large number of runs to be completed in a relatively short period of time.

Variable Name	*Scale*	*Default*	*Range*
Road Choice	N/A	N/A	0 - 4
Nominal Image Interval	Seconds	20	0.1 - 30
Data Transmit Interval	Seconds	2	1 - 10
Target Location Error (uniformly distributed +/-)	Feet	50	0 - 100
Target Velocity Error (uniformly distributed +/-)	Knots	3	0 - 3

Table 1. Simulation Variables

The model used several MATLAB® generated pseudo-random numbers to vary performance factors during each test case. MATLAB® contains a pseudo-random number generation system that produced a sequence of numbers determined by the state of the generator, and the initial seed. Since MATLAB® resets the state and seed at start-up, the sequence of numbers generated was the same unless the state or seed was changed.

C. WHAT DO THE RESULTS MEAN?

Based on any combination of inputs, and the state of each of the pseudo-random values embedded in the model, the output was a radial miss distance expressed as the difference between the calculated weapon impact point and the location of the true target in the 2-D plane. The representation of the weapon was limited to a simplified 2-D generic navigation model, so this miss distance does not truthfully represent the distance a weapon would miss the target.

In general the total miss distance is a function of the both the specific weapon accuracy and the ability of the intelligence or targeting system to locate the target correctly, and is expressed in a root sum square calculation as:

$$CEP_{Total} = \sqrt{CEP_{Weapon}^2 + CEP_{TLE}^2} \qquad (4.1)$$

To keep this research at the unclassified level, the simplified model used for this research sets the contribution of CEP_{Weapon} to zero[54]. This allowed the model to provide data for analysis that considered only the effects of various combinations of target uncertainty data inputs on the ability of the system to predict the future location of a target. The output of the model can be combined with the CEP_{Weapon} of any weapon using Equation 4.1, yielding a CEP_{Total}.

In defining requirements for a weapon system, the CEP_{Total} has a minimum acceptable value based on specific warhead and target interaction[55]. This research assumed that the contribution of CEP_{TLE} should be between 23 and 50 feet for the test case of road profile 'B' with variable velocity, to allow a variety of inventory weapons to be employed[56]. For any particular target as CEP_{TLE} is increased the weapon must be more accurate (i.e., the CEP_{Weapon} must be reduced). Figure 30 shows an example of the effect of increasing the CEP_{TLE}, assuming the allowable CEP_{Total} is 75 ft.

---

[54] The CEP_{Weapon} could be determined from either field test data or an accredited weapon model, and reflects contributions from weapon navigation system bias or errors (in three dimensions), weapon performance/maneuvering limitations, and ballistic dispersion of submunitions, if applicable.

[55] The Joint Technical Coordinating Group (JTCG) is chartered by the Office of the Secretary of Defense to collect, evaluate, and disseminate target vulnerability information. They publish the Joint Munitions Effectiveness Manuals which provide classified target/weapon specific data which can be used to determine the acceptable system CEP for that pair.

[56] The CEP_{TLE} for a horizontal, fixed target for the Joint Direct Attack Munition is specified as no greater than 7.2 m, approximately 23.6 feet. [JDAM ORD (Draft 13 Dec 99), p.5] For submunition-dispensing weapons, it is assumed to be a larger number due to the effective size of the submunition pattern versus the point warhead of JDAM.

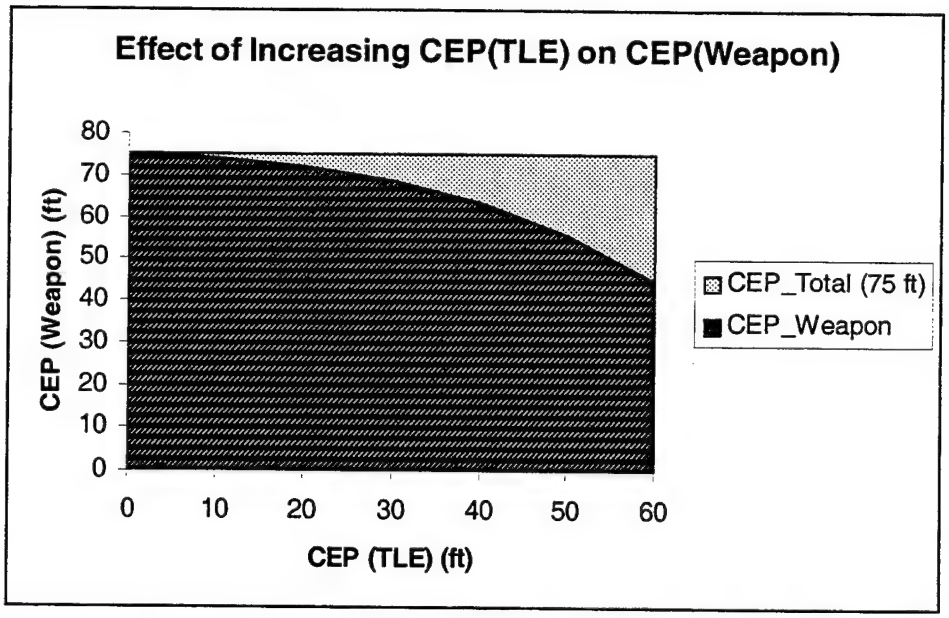


Figure 30. Effect of Increasing CEP_{TLE}

From the figure, the effect of Equation 4.1 is obvious. In this example, assuming a CEP_{TLE} of 50 ft., the maximum CEP_{Weapon} is 55.9 ft. If CEP_{TLE} increases to 60 ft., the maximum CEP_{Weapon} is decreased to 45 ft. Clearly the smaller CEP_{TLE}, the more flexible the strike planner can be with choice of weapon. If a particular weapon cannot achieve the necessary accuracy, another weapon may be selected by the strike planners with a warhead that is more effective against the desired target. Detailed analysis of individual weapon choices is considered beyond the scope of this research.

Finally, a limitation of the analysis was the number of data points obtained during each test case. While 100 data points for each run can be expected to give a general idea of the CEP[57] of the system, more samples are usually better. This limitation was based on processing and data analysis time available to compile the final 10,000 data points needed at this stage of the research.

[57] For the remainder of the paper CEP is used to mean the value in a set of sample data so that half of the data points are below the value, and half are above. Unless otherwise noted it is the 50^{th} data point in an ordered set of 100 data points from one test case run.

D. DATA ANALYSIS

Using the principle of superposition, combinations of input variables from the ranges in Table 1 were selected, which provided a spectrum of output. Variables not specifically discussed in any particular test set were set to the default values at the start of the simulation. The simulation was designed with the assumption that critical error sources would be traceable to data latency and target position and velocity accuracy, so the data developed was used to evaluate the effects of these factors. Table 2 shows the combinations of data that were collected for Road Profiles A and B, and these are discussed in paragraphs one through seven below. Note that there were some special cases run that are not shown in the table, but the rationale for these, and results, are discussed where appropriate. Subsection eight, below, will discuss the results of the data collected for the unconstrained motion case.

1. Perfect System

Setting TLE and TVE to zero, reducing the IPI and DTI to 0.1 seconds (the model had a code anomaly that didn't allow a value of 0.0 seconds to be used) represented the "perfect system". The results from a 'run' showed that the CEP was essentially zero (less than 1e-8) for all four road test cases.[58] This case is experimentally uninteresting except that it demonstrated the model appeared to be properly passing data to the weapon, and the weapon was reacting appropriately to target behavior.

[58] Constrained Random Motion was tested only as a single sample for this case. The radial miss distance from the one sample was 2.435e-7 ft. It is assumed additional runs would show the same result as the Road Profile test cases.

Road Profiles A & B		Target Location / Target Velocity Error			
Image Processing Interval		**0 / 0**	**25 / 3**	**50 / 3**	**100 / 3**
	0.1	X		X	
	5			X	
	10	X	X	X	X
	20	X	X	X	X
	30	X	X	X	X
Data Transmission Interval					
		1	**2**	**4**	**10**
	TLE/TVE	50/3	50/3	50/3	50/3
	IPI	20	20	20	20

Table 2. Variable Combinations Used During Data Collection

2. Relative Effect of IPI Latency

Setting the TLE and TVE errors to zero, and holding the DTI at 2 seconds isolated IPI. Figures 31 and 32 show the results of varying the IPI from 0.1 to 30 seconds. Clearly latency had a large impact on the effectiveness of the system. Comparing the results for the road profiles with and without target velocity variations there appears to be a direct correlation between increasing IPI and the relative magnitude of the resultant error when the target has a variable velocity.

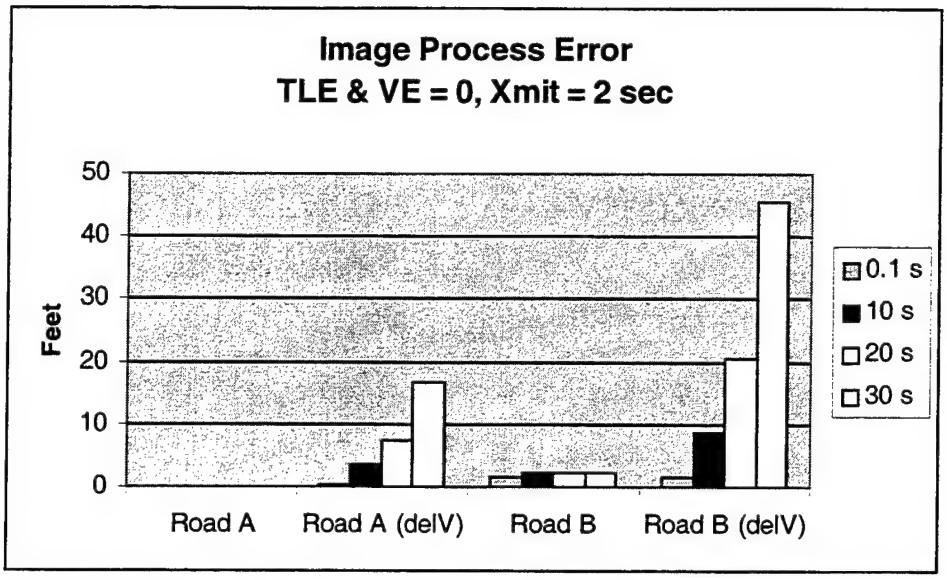


Figure 31. CEP Due to Image Process Time

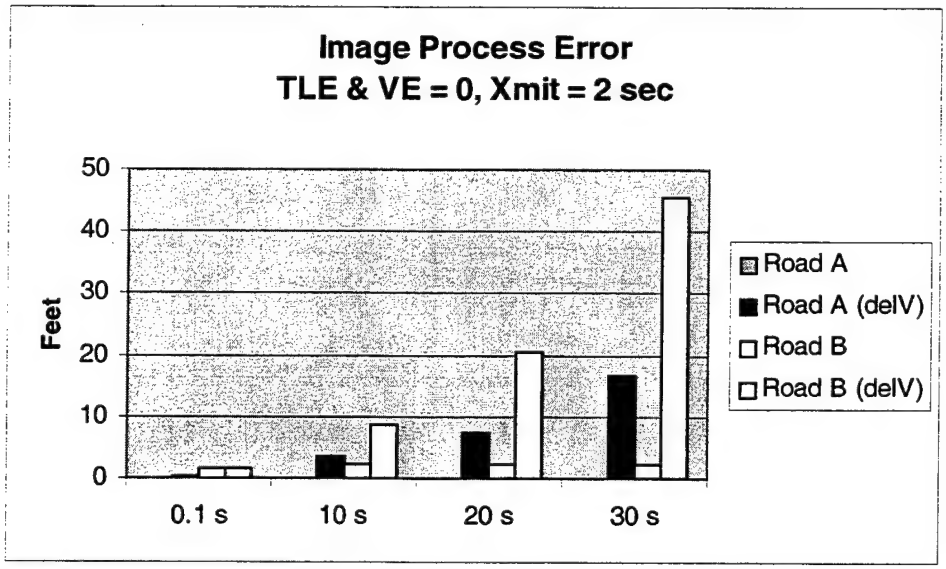


Figure 32. CEP Due to Image Process Time

3. Variable Image Processing Interval

While it is clear the better a system can locate a target the higher the probability the weapon will impact the target, assuming no TLE or TVE is unrealistic. It is useful to compare the effects at a more realistic value, although given a 'long' period of time to process imagery, one would expect TLE and TVE to approach zero. However, the latency introduced from this processing delay contributed to the overall system error. Figure 33 shows the effect of varying the IPI from 0.1 seconds to 30 seconds, while

holding other variables at the default values. While the statistical confidence in the absolute levels of the data is low, the relative levels show that increasing beyond a 10 second IPI raised the CEP above the previously stated acceptable threshold of 50 feet, and going to 30 seconds greatly increased the errors for all except the most 'cooperative' case.

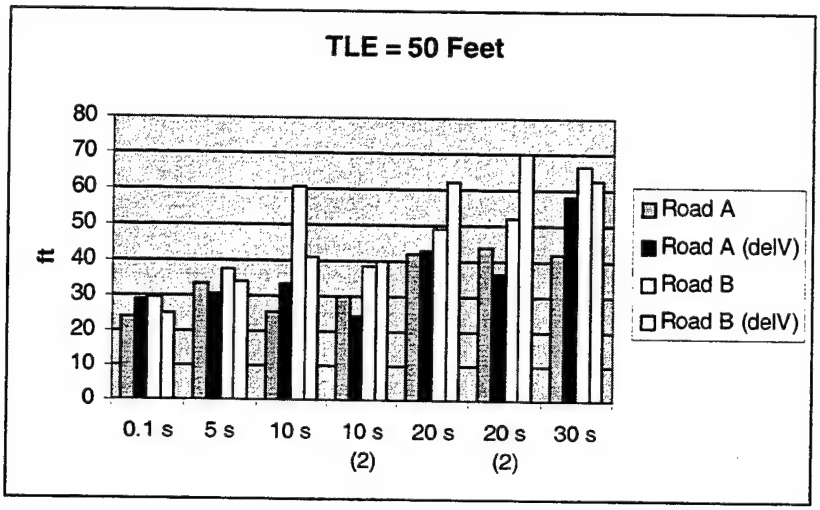


Figure 33. Relative Impact of Variable IPI[59]

Note the unexpectedly high peak in the 10-second data for Road B. This sample contained several Type-2 Gross Errors, which will be discussed later. The second 10-second data sample is more in line with expectation.

4. Imagery Induced Errors

Fixing the IPI at the minimum value of 0.1 seconds and DTI at 2 seconds isolated TLE and TVE. Figure 34 shows the results of varying the errors input into the system from the processing of the imagery. Note that for this test case when TLE was zero, TVE was also zero, and when TLE was 50 feet, TVE was 3 kts. This error source pairing was made by assuming that errors in position would correlate directly with errors in velocity. Thus even by providing a constantly updated image the combination of TLE and TVE contribute significantly to the error budget.

---

[59] The '(2)' indicates a second set of 100 runs made at the same conditions as a previous set.

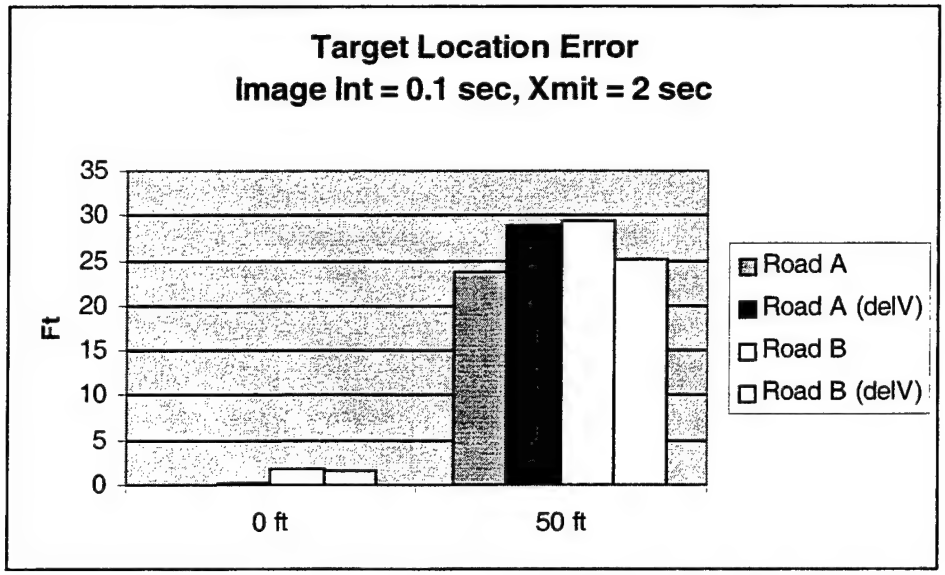


Figure 34. CEP Due to Target Location and Velocity Errors

In this case, and several others to follow, the final error is significantly less than the stated TLE, which may be confusing. A primary contributing factor to making the CEP 'better' is the a priori knowledge of the road vector, which helps remove the projection of the errors off the alignment with the road vector. As another contributing factor, TLE is input to the model as a uniform distribution in each axis of maximum size '+/- x' for any single model run. Each bar in a graph consists of 100 runs, so by the central limit theorem we would expect the distribution of all TLE inputs in either axis to approach a normal distribution with a mean of zero for the sum of the inputs. Note, however, the bi-variate distribution will tend to a non-zero mean since the total TLE is the square root of the sum of the squares of the individual axis inputs. Paragraph E will discuss the underlying distributions, and expected values.

5. Relative Impact of TLE and TVE

An obvious question was whether TLE or TVE had a greater impact on the error magnitude. In the model TLE had a direct impact on instantaneous position, and was a factor in the estimation of predicted positions for the data stream. The effect of TLE was somewhat mitigated in the prediction algorithm by ensuring the predicted locations were on the road, as shown in Figure 25. The effect of TVE showed up during calculation of predicted positions, as well. The decision not to change the target velocity during the data stream development allowed a known error to be allowed to propagate. The 3-knot

TVE continued over 60 seconds could result in an error of approximately 300 feet. However, excepting the unlikely probability of multiple sequential data dropouts in the transmission link prohibiting an update for a long period, most impacts should have occurred within the period of the data stream, so this error was in fact less in the simulation. With IPI and DTI at the default values Figure 35 shows the relative impact of TLE and TVE individually. From the graphs it appears these two components contribute relatively equally to the total error, and improving either would improve the overall capability of the system.

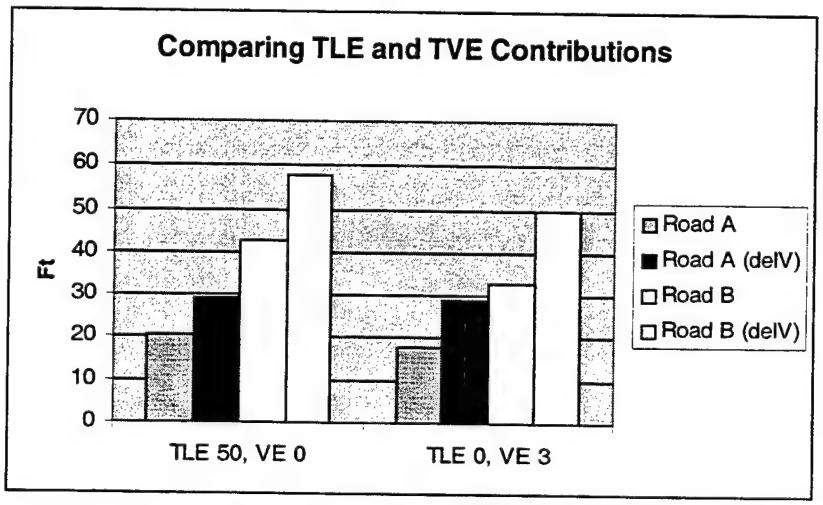


Figure 35. Relative Effects of TLE and TVE on CEP

6. Variable Target Location Error

As stated earlier, it is assumed that using more precise locating data will result in better overall performance better, and from the analysis of IPI it appears the system can accept a processing delay of approximately 20 seconds if needed to provide better precision. However, it is important to consider how 'poor' the locating data could be, yet still be acceptable for use. Figures 36 and 37 show the effect of increasing TLE from 0 to 100 feet with IPI at 10 seconds and 20 seconds, respectively. Note that at all TLE levels, except TLE of zero, the TVE was set to 3 knots based on the aforementioned assumption of correlation. Recall also the model uses the input value of IPI as a mean value and 'adds' a normalized pseudo-random value to it to determine the actual IPI for each iteration. From the figures it appears that for a nominal IPI of 10 seconds, a TLE of 50 feet would be acceptable, while for IPI of 20 seconds it will have to be slightly less.

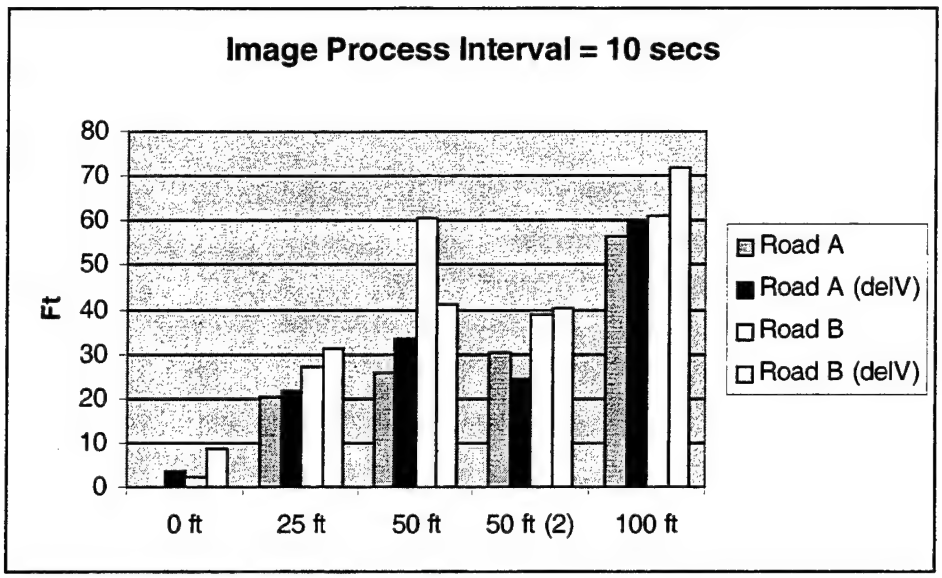


Figure 36. Variable TLE with constant IPI

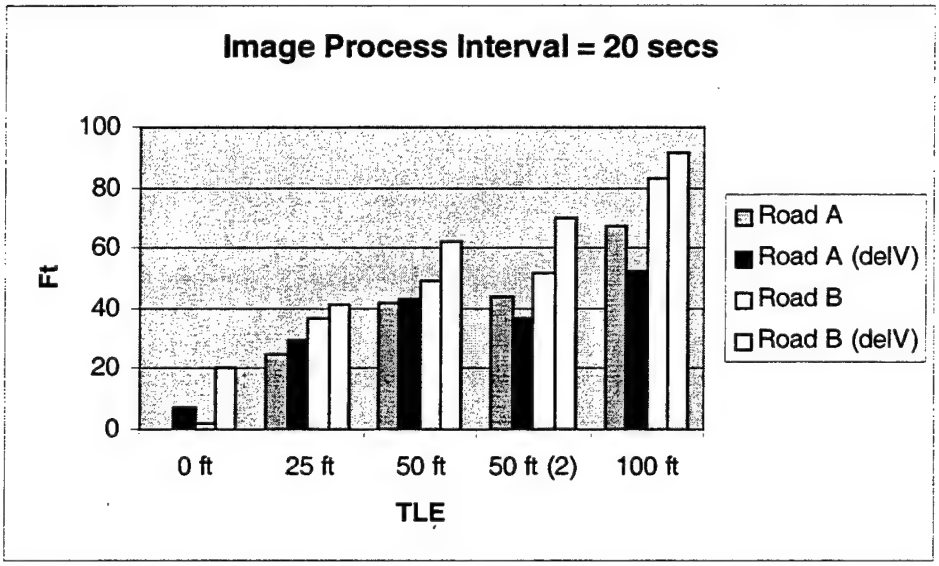


Figure 37. Variable TLE with constant IPI

7. Variable Data Transmission Interval

Data Transmission Interval is the regular (periodic) frequency at which the data was transmitted to the weapon. Simplistically the system could have sent continuous updates to the weapon in flight, which would define the minimum interval, or only send an update after a new 'image' was processed, which would define a maximum interval.

The minimum interval is effected by reasonable time slots and bandwidth available on the proposed Link-16, and by a desire to enable simultaneous control of ten

or more weapon/target pairs, each with its' unique data stream. At the lower end, the minimum interval would be ten times the length of the data stream, plus a small amount of overhead, or roughly 0.2 seconds. Considering the maximum interval, transmitting only when an update is completely processed would have minimal impact on other users of the link, but practically would not be acceptable due to the possibility of a transmission loss due to attenuation, relay antenna positioning, or loss of weapon line of sight.

To ensure the weapon has a frequent opportunity to receive the data, accounting for occasional dropouts in the transmission link a fixed, periodic, interval is desirable. Figure 38 shows the effect of changing the DTI from 1 second to 10 seconds. Note this was a periodic update, and was not synchronized to data extraction from imagery. From the figure the accuracy appears relatively unaffected out to a 2 second DTI, and is not completely unacceptable until beyond 4 seconds. The model did have a random data dropout rate in the transmission module of 2%, so it is reasonable to assume the runs shown include on average two data interruptions each which increased the transmission interval to at least one and perhaps multiple periods. Figure 39 shows the same data grouped by road profile.

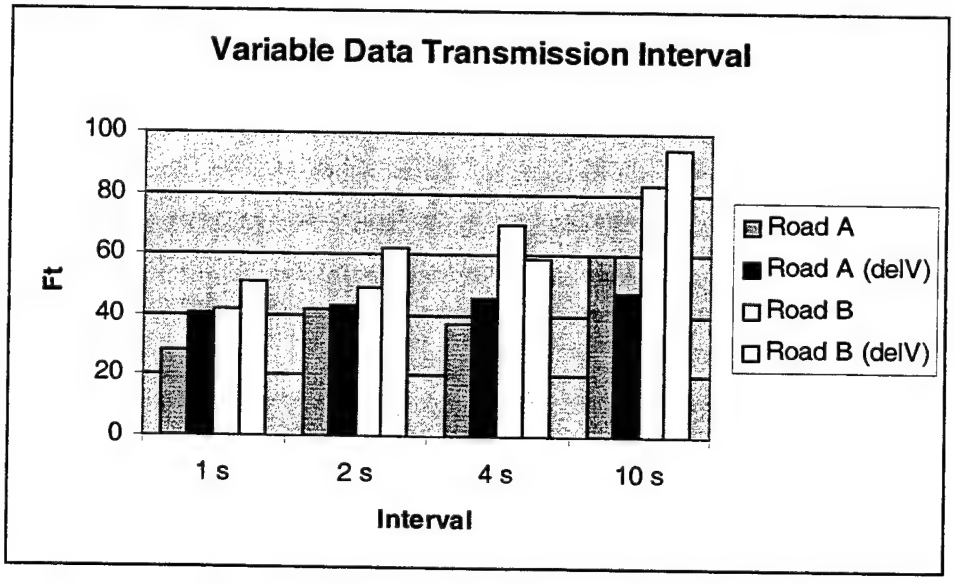


Figure 38. Variable Data Transmission Interval

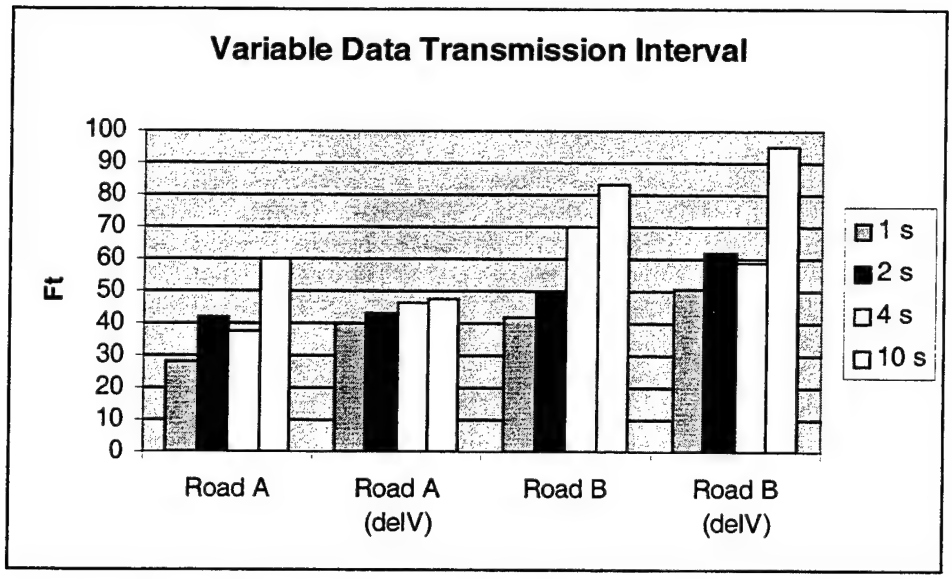


Figure 39. Variable Data Transmission Interval

8. Effectiveness Against a Constrained Random Motion Target

In this case no prediction of future target motion could be made, so the test hypothesis was the critical factor would be the ability of the system to rapidly produce target coordinates from the sensor data. Under this assumption the variable selected for analysis was the IPI. Several runs were made against the constrained random motion target model using the default values (DTI = 2s, TLE = 50 ft, TVE = 3 kt) shown during the analysis to be relatively effective against the primary test case of road profile 'B' with variable velocity. Using a minimum value of 5 seconds, and a maximum of 20 seconds, the results are shown in Figure 40. From the figure the effectiveness appears to be marginal. Somewhat surprisingly there is little effect from decreasing the IPI below 10 seconds. Whether this is due to the simplicity of the model or is a true limitation is uncertain, and is left for future study.

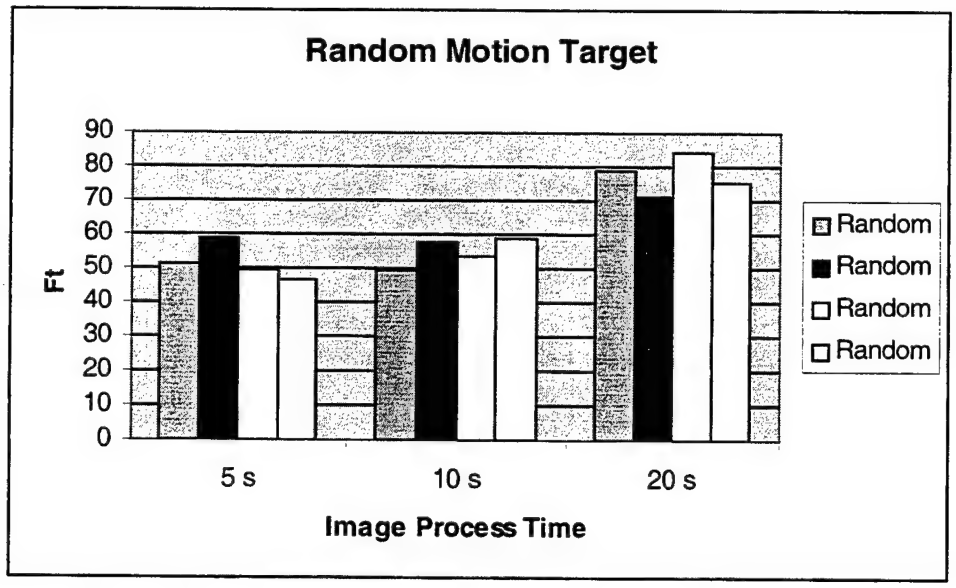


Figure 40. Effectiveness Against Random Target Motion

E. DATA VALIDITY

With results in hand, the question is: are the results meaningful, or are they are just pretty pictures? Ideally the output of the model would be validated with a comparison to test cases using an actual system, and the model would be tweaked to make it perform properly. This process of validating the model by ensuring the product satisfies functional requirements was impractical at this phase of development, since there was no real system to compare the results, and the model was designed to operate only at a top level with inherent simplifications. Instead, the model was only verified to ensure it performed the functions it was intended to perform.

1. Missile Behavior

The missile behavior model was designed to perform a lead angle intercept of a moving target. The simplest test case is road profile 'A' with fixed target velocity and no TLE or TVE. If the model was working properly, the weapon should show a maneuver to remove the initial heading error, and stabilize on a straight-line trajectory to an intercept point. Figure 41 shows this case meets the expected performance. If the weapon behavior were incorrect the intercept track would have been continually concave, in the case of a pure pursuit or lag-pursuit, or would have shown numerous intermediate corrections.

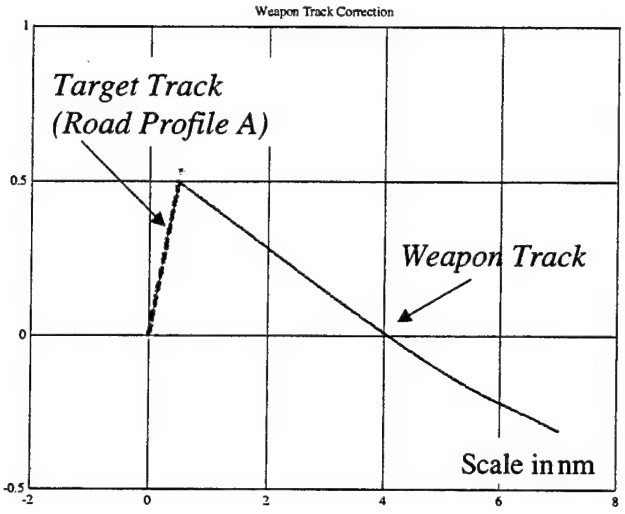


Figure 41. Typical Weapon Track to Intercept without Targeting Error

The next test case was to analyze the same road profile, but with TLE and TVE set to the default values. Figure 42 shows the weapon still performed as expected, although the TLE and TVE resulted in some intercept uncertainty.

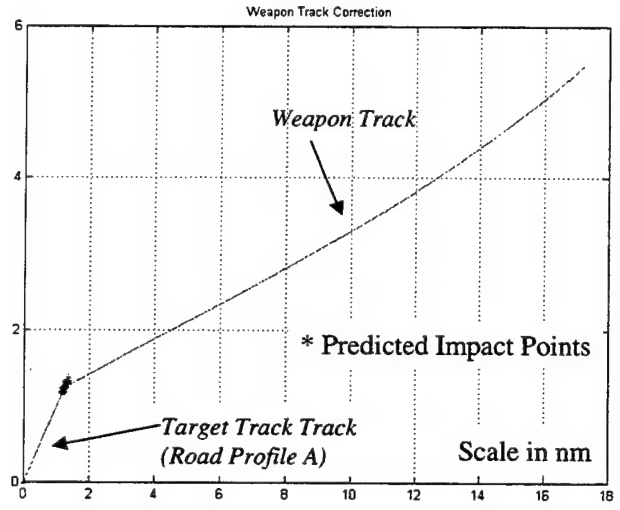


Figure 42. Typical Weapon Track to Intercept with Targeting Error

The final test case used road profile 'B' with variable target velocity, as well as TLE and TVE set to default values. Figure 43 shows the weapon track made an initial correction, and in this case also made subsequent corrections at the times corresponding to the target approaching the turns. These corrections show adjustment for target changes

in direction as well as target velocity. Based on these test cases, the missile behavior model appears to performing the proper lead computation.

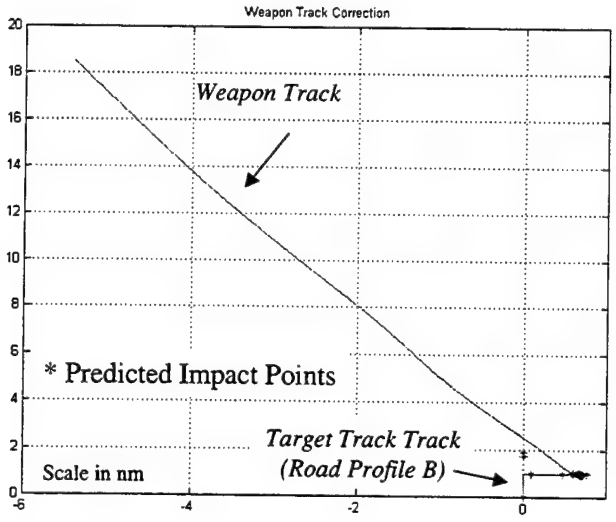


Figure 43. Typical Track to Intercept with Targeting Error

2. Statistical Analysis

Since the model appeared to be working properly, the next question was whether the model output was statistically consistent with expectations. The usual starting point when analyzing a distribution of impact points from weapon testing is to assume the miss distances in the range (M_r) and deflection (M_d) directions are normally distributed about the DMPI with equal variances and a mean of zero. Using the assumption of normality the mapping function for radial miss distance (RMD)[60]

$$RMD^2 = M_r^2 + M_d^2 \qquad (4.2)$$

is a circular normal distribution where, for each (M_r, M_d) pair, there is a single value of RMD, and this value must be equal to or greater than zero. Thus RMD cannot be normally distributed, but instead will have a Rayleigh distribution.

In practice, however, it is not unusual to find during analysis of the data the observed RMD distribution is not circular, due to non-equal variances in the underlying M_r and M_d distributions. The output of the model was not initially structured to resolve

[60] The model assumed a zero contribution from the weapon to the overall CEP, the RMD reflects only the effect of targeting error.

the miss distances in the range and deflection directions; rather a singular output of the RMD was presented. Applying the Central Limit Theorem to the model design it was reasonable to accept as true the hypothesis that M_r and M_d were normally distributed. However, it was not obviously true that they had equal variances, thus the validity of the expectation of a Rayleigh distributed RMD population was unknown. To test this hypothesis, the model was modified and run separately to produce one set of range and deflection error data for M_r and M_d, which required resolving the radial miss distance into orthogonal components relative to the weapon heading as shown in Figure 44.

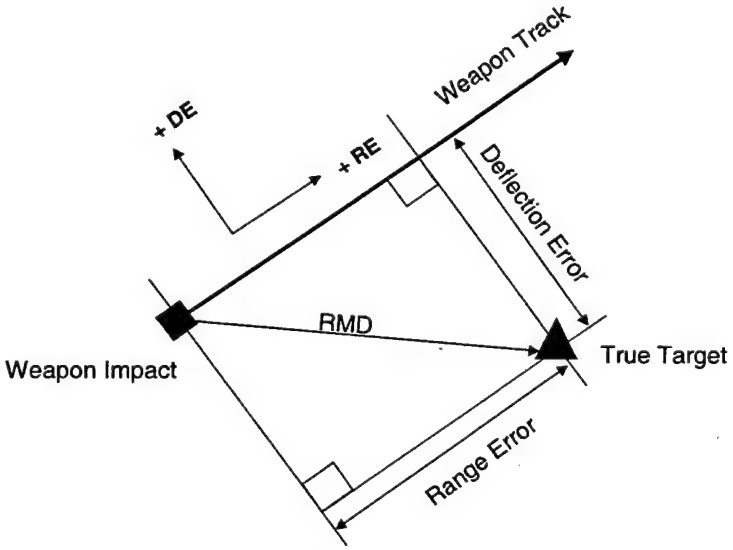


Figure 44. Resolving RMD into Range and Deflection Components

A histogram of the data from road profile A, with all variables set to the default values, is shown in Table 3. The histogram also shows the expected distribution using data from a normal probability density function. To measure how close the data was to the expectation, a Chi-squared analysis was made, and the result showed only a 27.36% probability that the range errors are normally distributed, but an 88.63% probability that the deflections errors are normal. Upon closer examination of the data it was apparent the frequency of data from both distributions near the mean was significantly higher than the expected value. The range error distribution is shown graphically in Figure 45. This

relatively large number of "direct hits" is characteristic of guided weapons[61], and should have been expected from this system.

Range Error

Bin	Frequency	Expected Normal RE
-40.00	2	2.75
-30.00	4	4.61
-25.00	3	3.86
-21.60	4	3.34
-17.85	5	4.40
-14.10	6	5.13
-2.27	20	20.02
2.52	18	8.94
8.20	10	10.37
13.50	7	8.88
17.75	5	6.26
22.00	3	5.35
27.50	3	5.50
31.80	1	3.24
33.60	1	1.11
48.88	5	5.02
59.44	2	0.91
63.72	1	0.14
More	0	0.16
	Chi Test	27.36%

Deflection Error

Bin	Frequency	Expected Normal DE
-200.0	1	0.94
-176.0	2	1.00
-169.0	1	0.42
-135.0	3	3.28
-120.0	2	2.29
-98.0	2	4.54
-63.0	6	10.46
-42.7	9	7.82
-20.0	11	9.89
1.5	14	9.97
20.0	11	8.57
39.9	9	8.74
54.0	8	5.67
90.0	8	11.79
100.0	2	2.52
120.0	2	4.09
140.0	3	2.95
160.0	2	2.01
165.0	0	0.38
230.0	2	2.31
More	1	0.35
	Chi Test	88.63%

Table 3. Histogram of Raw Range and Deflection Data

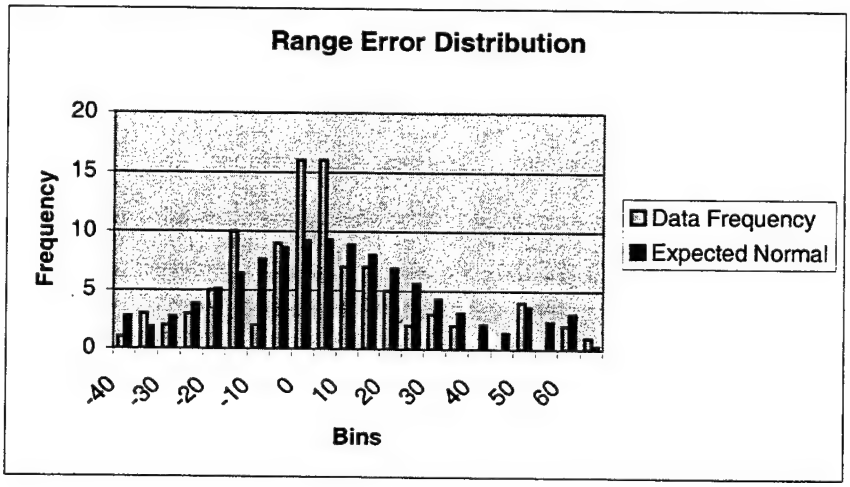


Figure 45. Range Error Distribution

[61] Driels, *Weaponeering*, pp 3-34. Although there is no weapon contribution, this result should have been expected since the periodic input from the sensor produces the same effect on the targeting error.

Removing some of the 'excess' direct hits from the data, the revised histograms are shown in Table 4, and show a significantly higher confidence that the data distributions are indeed normally distributed.

Range Error

Bin	*Frequency*	*Expected Normal RE*
-40.00	2	2.75
-30.00	4	4.61
-25.00	3	3.86
-21.60	4	3.34
-17.85	5	4.40
-14.10	6	5.13
-2.27	20	20.02
2.52	9	8.94
8.20	10	10.37
13.50	7	8.88
17.75	5	6.26
22.00	3	5.35
27.50	3	5.50
31.80	1	3.24
33.60	1	1.11
48.88	5	5.02
59.44	2	0.91
63.72	1	0.14
More	0	0.16
	Chi Test	84.98%

Deflection Error

Bin	*Frequency*	*Expected Normal DE*
-200.0	1	0.94
-176.0	2	1.00
-169.0	1	0.42
-135.0	3	3.28
-120.0	2	2.29
-98.0	2	4.54
-63.0	6	10.46
-42.7	8	7.82
-20.0	10	9.89
1.5	10	9.97
20.0	9	8.57
39.9	9	8.74
54.0	8	5.67
90.0	8	11.79
100.0	2	2.52
120.0	2	4.09
140.0	3	2.95
160.0	2	2.01
165.0	0	0.38
230.0	2	2.31
More	1	0.35
	Chi Test	96.45%

Table 4. Histogram of Range and Deflection Data with 'Direct Hits' Removed

Table 5 shows the mean and standard deviation for this sample data. The means are sufficiently close to zero to accept the hypothesis the system is unbiased, however the standard deviations are significantly different, by a factor (smallest divided by largest) of 0.305. Although the RMD is not a Rayleigh distribution due to the different underlying distributions it would be acceptable practice to use the circular case for non-circular data, when the ratio of standard deviations is no less than about 0.5. Since the data sample falls below this threshold, for smaller ratios where:

$$0.2 < \frac{\sigma_s}{\sigma_l} < 0.5 \qquad (4.3)$$

the following equation may be used[62]:

$$CEP = 0.8728(REP + DEP) \quad (4.4)$$

The REP and DEP were manually calculated from the sample by taking the absolute value of each and selecting the 50th element of the list, which resulted in REP = 10.29 ft, and DEP = 44.35 ft. Applying Equation 4.4, the predicted CEP is:

$$CEP = 0.8728(10.29 + 44.35) = 47.69\,ft$$

This is about 17% higher than the model value of 40.78 ft., but within acceptable tolerance for a single sample to postulate agreement.

Finally, if the mean miss distances are zero, using a polar coordinate transformation there is a form of the bi-variate normal distribution that is numerically solvable, and which may be approximated by Pittman's equation[63]:

$$CEP = 0.562 * MAX(\sigma_{rep}, \sigma_{dep}) + 0.617 * MIN(\sigma_{rep}, \sigma_{dep}) \quad (4.5)$$

Using the values from Table 5 in Equation 4.5:

$$CEP = 0.562 * 69.94 + 0.617 * 21.31 = 52.45$$

which is a larger deviation from the sample data. Additional study can be made, if this model is further developed, to ascertain the degree of agreement using more samples.

Probability Analysis of Raw Data

	Range Error	Deflection Error
Median	-0.27	-4.25
Mean	0.89	0.20
Standard Deviation	21.31	69.94

Table 5. Probability Analysis of Raw Data

Finally, the data obtained from the model was compared graphically to the values that should be expected from a Rayleigh distribution, as well as to the distribution of expected values using Equations 4.4 and 4.5. The results are shown in Table 6, and the distributions are graphically compared in Figure 46.

---

[62] Driels, M., *Weaponeering*, pp. 3-6.

[63] Driels, M., *Weaponeering*, pp. 3-7.

Bin	*Data Frequency*	*Expected Rayleigh*	*Equation 4.4 Formula*	*Pittman's Formula*
8.87	2	2.84	1.70	1.29
19.12	12	9.63	5.95	4.56
35.04	18	23.06	15.59	12.35
52.24	19	25.19	20.51	17.38
65.45	13	14.10	14.72	13.64
79.17	9	8.91	12.33	12.65
94.80	8	4.84	9.68	11.36
106.00	4	1.45	4.39	5.96
137.43	4	0.94	5.15	8.87
More	2	0.04	0.97	2.94
	Chitest	0.00%	48.91%	2.13%

Table 6. Histogram Comparison of Distributions

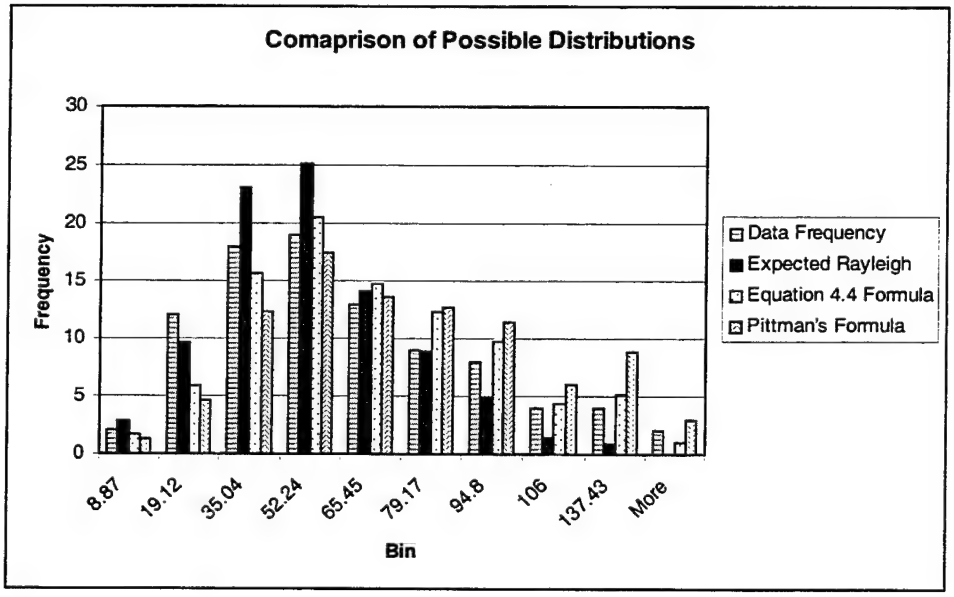


Figure 46. Graph of Distributions from Table 6.

Table 6 shows a 49% probability the sample agrees with the expected distribution using Equation 4.4, but one sample is insufficient to make any final conclusions on the actual distribution.

3. Sample Size

As noted earlier, each data run resulted in 100 data points for each of the four road/velocity pairs, and it was stated that this sample size was representative of the true population. To test this hypothesis, one set of data runs was made to provide 1000 data points for each of the four road/velocity pairs. The resulting CEP, as well as the mean and standard deviation of the sample set, compared favorably to those characteristics of the smaller sample sizes using the same variable combination. A comparison of the CEPs is shown in Figure 47.

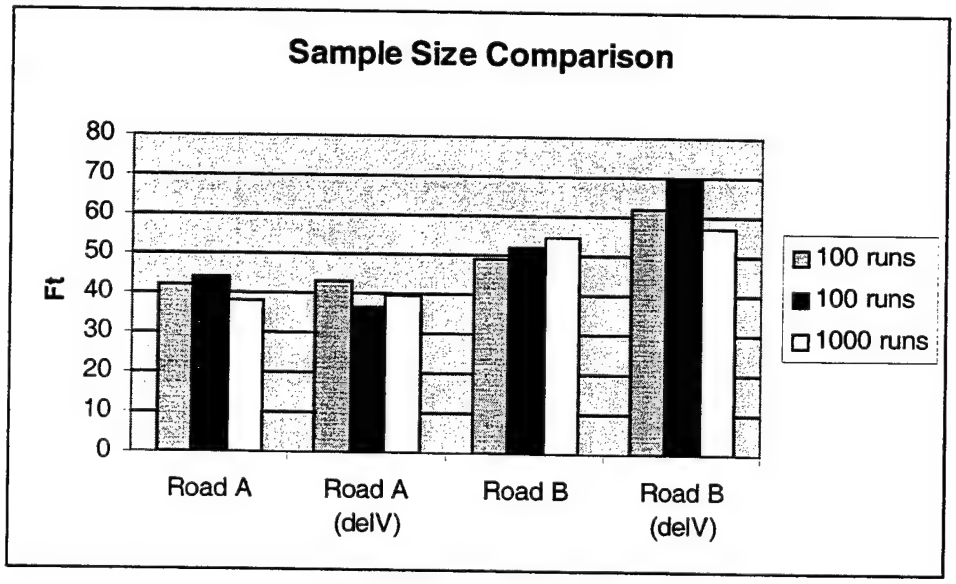


Figure 47. Sample Size Comparison

4. Gross Errors

The output of any run of the model contained several miss distances which were classified as 'gross errors' if they were more than 4σ from the sample mean. While the nature of the test runs made it impractical to analyze each of these gross errors, during model development and testing two contributing factors were noted which explain some, if not most, of the occurrences.

A Type-1 'gross error' was characterized by an RMD at least two orders of magnitude greater than the CEP. This error occurred in only 0.11% of data points observed. The cause of this error can be traced to a run when the weapon-to-estimated target separation, as calculated in the "Missile Behavior Subsystem", did not enter the 0.1

nautical mile circle around the estimated target position. In this case the 'stop sequence' was initiated at 4.9 elapsed 'minutes' to ensure the simulation did not read past the end of the target motion data file. The weapon/target separation at this point in the simulation was obviously large. In a 'real' weapon system such hits are unlikely to occur since the weapon is flying to an impact point. Therefore type-1 gross errors are not considered part of the representative statistical population. The model was not modified to reduce the probability of this error because of the low observed occurrence.

A Type-2 'gross error' includes the RMDs that were greater than 4σ from the sample mean after Type-1 'gross errors' were removed from the sample. This error was observed on only 1.09% of the data runs and, significantly, was not limited to data obtained with any particular input variable setting. For example, with all variables set to minimum values resulting in a 'near perfect' system, there were still eight type-2 'gross errors' in the data, while in the data using a TLE of 50 feet with a DTI of 10 seconds, there were seven type-2 errors. The distribution of type-2 errors by road profile is shown in Figure 48.

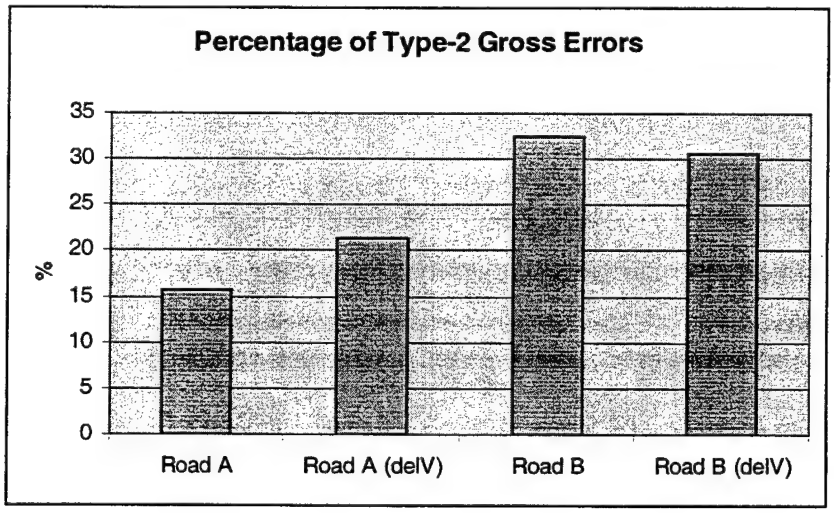


Figure 48. Percentage of Type-2 Gross Errors by Road Profile

The individual cause of the error cannot be ascertained in each case, however looking at the distribution above some of the probable causes can be speculated. The largest percentage of these hits likely involved simulations which resulted in an impact time near the points when the target made heading or velocity changes, for example near the turns, which necessarily resulted in an incorrect lead distance. This is a direct result

of model simplification, and the real world system would likely include a better velocity estimation algorithm to account for these cases. Another possible cause was observed on many test runs when the 'stop simulation' command was not executed immediately, resulting in multiple (as many as four were observed) "end" statements being displayed in the MATLAB® 'command window'. Through empirical testing it was determined that in this case only the final RMD displayed was being stored as the result. Usually the deviation between RMDs associated with each "end" statement group was relatively small, however in some cases they were significantly different (e.g., 349.212 feet followed by 879.3619 feet). While the resulting values were not limited to inclusion in only gross errors, and were not always the largest of the values from a particular "end" statement group, this was an error source which was observed to contribute to a small percentage of type-2 errors. Hits resulting from type-2 'gross errors' are judged to be primarily due to model inaccuracies. However, since similar causes of these errors can occur in a real system they were considered as part of the representative statistical population.

V. CONCLUSIONS AND RECOMMENDATIONS

A. SUMMARY OF RESULTS

This research developed a requirements based model of a sensor-to-weapon system to provide constantly updated target location information to a GPS-guided weapon after launch. The model was used to evaluate requirements for a system using a GPS-aided low cost conventional standoff weapon to attack time sensitive moving targets without the aid of a weapon-based terminal seeker. The variables considered during this requirement analysis were combinations of Target Location Error, Target Velocity Error, Road Profile, Image Processing Interval, and Data Transmission Interval. The results of the model analysis show that this type of system is technically feasible without overly restrictive requirements. Assuming the contribution of targeting system error to the overall system error budget is CEP_{TLE} = 50 ft, the research showed that the performance requirement limitations shown in Table 7 would allow the system to target with acceptable accuracy.

Critical Parameter	***Scale***	***Acceptable Value***
Nominal Image Interval	Seconds	< = 20
Data Transmit Interval	Seconds	< = 2
Target Location Error (uniformly distributed +/-)	Feet	< = 50
Target Velocity Error (uniformly distributed +/-)	Knots	< = 3

Table 7. Critical Technical Parameters

Using this value of CEP_{TLE}, Figure 49 shows the acceptable range of CEP_{Weapon} for a range of CEP_{Total}, using Equation 4.1, which is repeated here for convenience.

$$CEP_{Total} = \sqrt{CEP_{Weapon}^2 + CEP_{TLE}^2} \qquad (4.1)$$

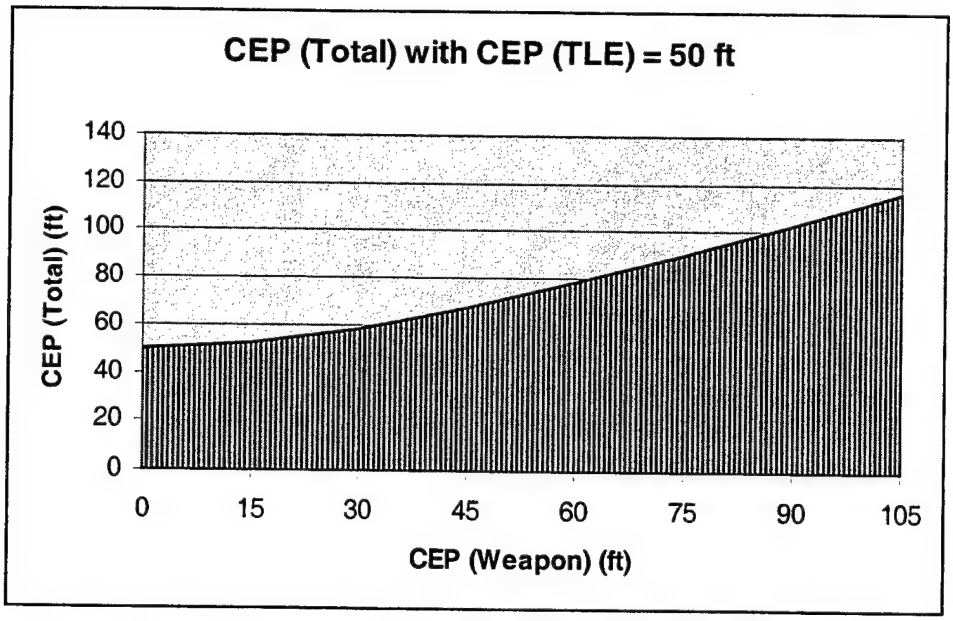


Figure 49. CEP_{Total} with Increasing CEP_{Weapon} and Fixed CEP_{TLE}

B. TACTICAL CONSIDERATION

The author is keenly aware of the dangers of addressing tactical employment considerations, however one observation is in order based on the data analysis conducted in this study. The success of an engagement using this system is dependent on the ability to predict future target movement. Planning for an engagement to occur on a predictable section of a road will improve the probability of success.

C. RECOMMENDATIONS FOR FUTURE RESEARCH

- The model developed for this research intentionally used unclassified data parameters that reduced the absolute accuracy of output data. The use of classified data on an appropriately classified processing system will improve the fidelity of the output.

- The model was designed with modular components to allow for future upgrades. Improved definition of the "Sensor/Tracker Subsystem" to include target motion prediction algorithms would enhance the ability of the system to show sensitivity to target motion changes. Specifically, establishing the data stream intermediate data points by incorporating a velocity change prediction model to estimate position using non-linearly spaced position points will

improve the accuracy of the system. Keeping the spacing of the data points at 1-second intervals will minimize the content of the data stream.

- The input errors for TLE and TVE were based on a uniform pseudo-random variable at each time step. Analysis of the output of the actual, or a representative, position derivation system would allow a more accurate probability density function to be incorporated into the model.
- The weapon behavior model is generic. Incorporation of a 3-DOF kinematic model of a JSOW or SLAM would allow the inclusion of the CEP_{Weapon} in the model response.
- To evaluate the utility of the TSMTS, 'Use Cases' should be created which describe in a very generic sense the desired activity of the system. Performance-based cases should reflect typical tasks that the TSMTS may be called on to execute. As part of the research, four 'Use Cases' have been tentatively identified for TSMTS:
  1. Track a single target, and provide information to a single weapon.
  2. Track a single target, and provide information to more than one weapon.
  3. Track a series of targets that correlate their movement to follow a lead target (e.g., a convoy), and provide information to multiple weapons.
  4. Track a series of targets that have uncorrellated movement, and provide information to multiple weapons.
- The utility of this system is not limited to seekerless weapons. Including a seeker model in the weapon behavior subsystem to allow terminal engagement will allow the consideration of larger system error tolerances.
- The length of the data stream was fixed at 60 seconds during this research. Consideration of algorithms to optimize this length based on expected maneuvering capability of the target may produce a different value. Similarly the inter-data point spacing was set at 1 second. Decreasing the interval may have some effects that were not anticipated during this research.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. MODEL USE

A. SET UP

The source files used for this research may be requested from Professor R. Duren, Department of Aeronautics and Astronautics, Naval Postgraduate School. The SIMULINK® models and MATLAB® m-files must be located in the same directory. If the target movement files are not available, the SIMULINK® model "Roadtest.mdl" should be run first to generate target motion data for each road profile and stored as *.mat files.

B. DEFAULT MODEL USE

To run the model in the default mode, accepting default values, the User need only initiate the Simulink® model. While the system is running, the model will display three motion windows showing the missile motion, target true motion, and target 'dr' motion respectively. These windows may be moved as desired by the User. When the simulation completes, the MATLAB® command window will display the calculated radial miss distance (labeled CEP) and the estimated lethality of a generic warhead against a generic target (labeled SSPD). Subsequent runs of the model, assuming MATLAB® is not reinitialized, will produce different missile initial positioning, yielding different engagements. If MATLAB® is restarted, the first run initial position can be replicated.

C. MODIFYING DEFAULT VALUES

To modify the default values, the User should open the "Input and Errors Subsystem" window. Figure A-1 shows the seven primary variables that may be changed for any run.

1. Type road: The variable "type road" has acceptable integer values from [0..4], and results in a data file extraction as annotated in Figure (A-1).

2. TLE & V_err: The TLE and TVE output is composed of four bussed variables that may be manipulated in several combinations. The gains shown in Figure A-1 represent switches to turn on/off any individual or set of variables. To turn a variable off, the respective gain should be set to '0', to turn it on it should be '1'. The 'Gain4' block may be used to turn on/off all four variables at once.

The individual variable characteristics may be adjusted within the respective blocks. The TLE (in both x and y) is modeled as a Uniform Random variable with a maximum of +/- 50 feet. The actual input in the block must be '50/6075' since distances in the model are scaled to nautical miles. To change from '50 feet' to any value 'R feet', the input should be in the form 'R/6075'. The TVE (in both x and y) is modeled as a Uniform Random Variable with a maximum of +/- 3 kts. The input is in the form '3/60' since velocities in the model are scaled to nautical miles per minute. To change from the value '3 knots' to any value 'V knots' the input should be in the form 'V/60'.

3. Image Processing Rate: The image-processing rate is the nominal period the model will use to 'hold' image data before releasing it to the transmission subsystem. The input must be in the form 'T1/60' where T1 is the number of seconds of delay desired. The minimum acceptable value tested was '0.1/60', or one-tenth second delay. Note that when running at this interval the system response is extremely slow due to the continual update of target estimation data. Maximum value tested was '120/60'.

4. Data Xmit Rate: The data transmission rate is the nominal interval at which data updates sent to the weapon receiver. The input must be in the form 'T2/60' where T2 is the number of seconds delay between sequential data transmissions. The minimum acceptable value tested was '0.1/60', or one-tenth second delay. Maximum value tested was '10/60'.

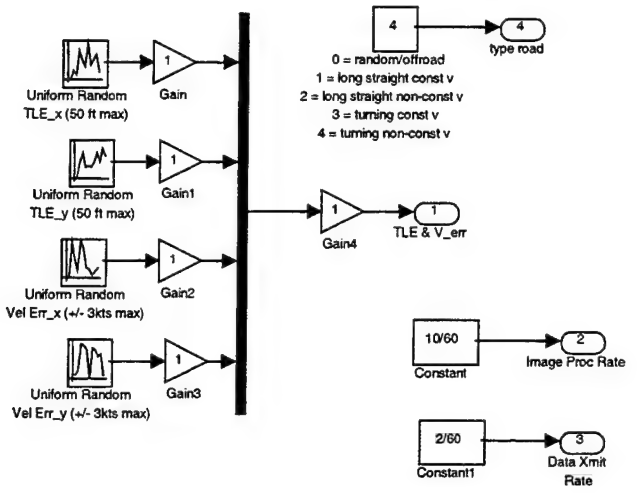


Figure A-1. Primary Input Variables

D. INTERNAL CHANGES

The model contains several variables that may be manipulated to modify the performance either by modifying an m-file called by the model, or one of the probability functions used within the model. Although the author believes the code to be reasonably well documented, caution should be used when modifying any file. The models and source code for files referenced below may be found in Appendix B and C.

The Top Level model has a constant input into the "Lethality Subsystem" which is set to '1'. This represents the number of weapons to be used in the calculation of the SSPD. This value may be increased to obtain the effects from multiple weapons on the same target, although as currently modeled each weapon will have the same miss distance, which is unrealistic.

The model "Target Vel Vector" in the "Target Truth Subsystem" uses a normal random variable to affect the axial acceleration for the constrained random motion case. The variance of this block can be adjusted to change the acceleration characteristics of the target.

The model "Missile Navigation Subsystem" in the "Missile Behavior Subsystem" uses two normal random variables to input the wind effects on the x- and y-axis of the missile. The variance of these blocks may be adjusted to change the affects of the wind on the missile.

File "if_stop.m" called by "Missile Behavior Subsystem" controls the simulation stop parameters. These are coded at 1/10-mile radius and 4.9 minutes, as noted in Chapter III.

File "latch.m" called by "Transmission Subsystem" uses a MATLAB® pseudo random number generator to initialize the variable 'skip' which controls the probability the scheduled data transmission will be skipped. The characteristics of the random number generator may be adjusted, or the probability may be changed from the default value of 0.98.

File "latch_image.m" called by "Image Processor Subsystem" uses a MATLAB® pseudo random number generator to initialize the variable 'r_proc' that controls the variability of the image processing time. 'r_proc' is added to the value input through the "Image Processing Rate" shown in Figure A-1.

File "Pk.m" called by "Lethality Subsystem" contains generic weapon effect data and may be adjusted. If desired, the User is recommended to reference JMEM methodology and is cautioned to consider classification of data.

File "start_pos.m" called by "Missile Navigation Subsystem" controls the initialization of the missile start position relative to the origin of the x-y plane. The model is currently limited to not more than 20 NM in either axis, but at least 5 NM in a straight line from the origin. These values may be changed to allow either a larger area, or a closer launch range. Note that if a larger area is enabled, the User should consider the amount of target motion data available in the file, as well as the simulation stop conditions.

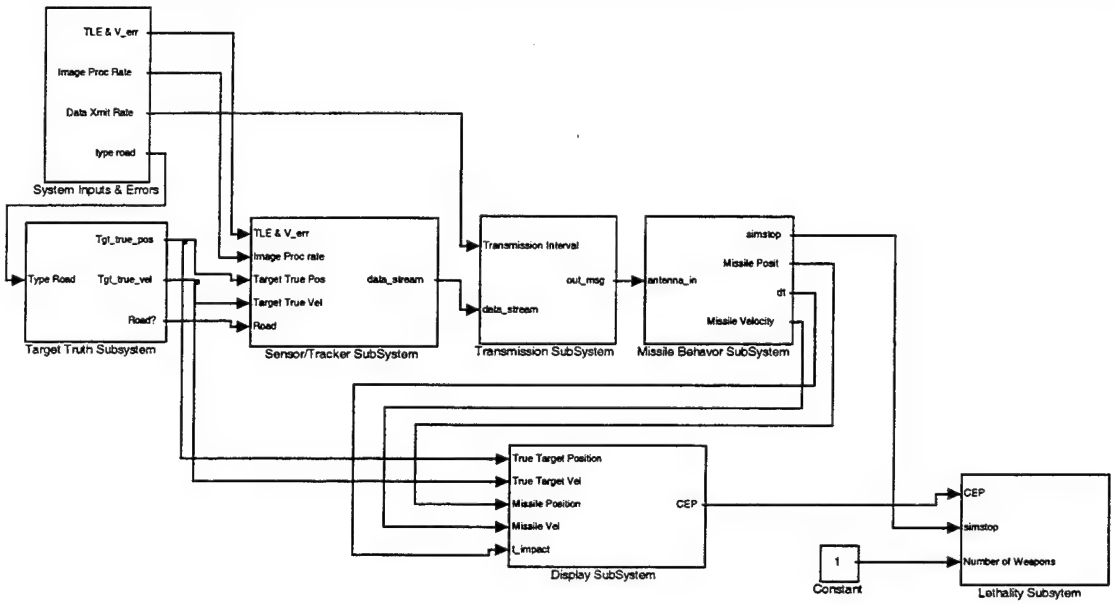
E. MULTIPLE SEQUENTIAL RUNS

To facilitate unmonitored sequential runs of the model while varying desired parameters a MATLAB m-file was written to interface with the model. This file requires the model to be slightly modified from the single-run settings that are shown in Appendix B, however these modifications are described in the header and comments of the file "test.m" which can be found in Appendix C. When implemented, the model will run repeatedly with no User input required, and save data to workspace files for later review.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. SIMULINK® MODELS

Top Level



Brief Description:

Within the model the system is partitioned by functionality to mirror the proposed system described in chapter 2, with the main functional objects being the "Target Truth Subsystem", the "Sensor/Tracker Subsystem", the "Transmission Subsystem", and the "Missile Behavior Subsystem." For data input and analysis three additional subsystem components were added: "System Inputs and Errors", "Display Subsystem" and "Lethality Subsystem."

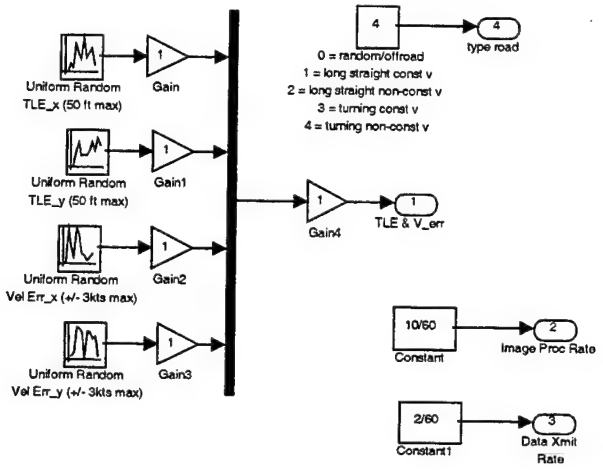
References:

None

M.file File Headers:

None

Input and Errors Subsystem



Brief Description:

This subsystem block was used in lieu of a graphical user interface to adjust the primary simulation variables during the simulations. The variables include target location error, target velocity error, image processing rate, data transmission rate, and movement scenario during simulations.

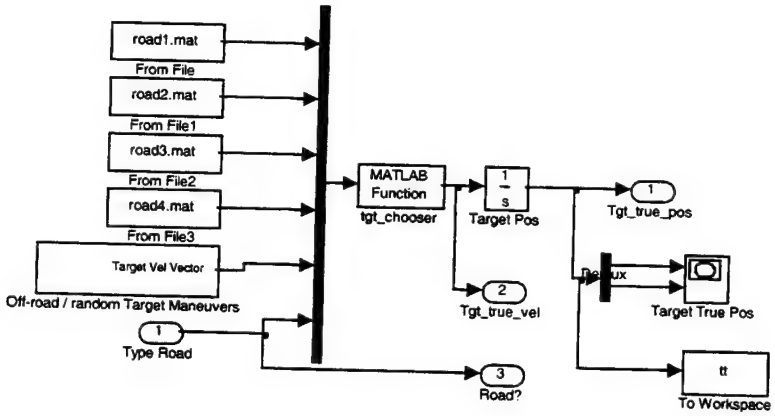
References:

None

M.file File Headers:

None

Target Truth Subsystem



Brief Description:

The "Target Truth Subsystem" controls target movement during the simulation, and provides an output of true target position for analysis of accuracy. The target is a single vehicle moving at a nominal 45 knots in the 2-D ground plane. Target motion data is input to the model from stored data files.

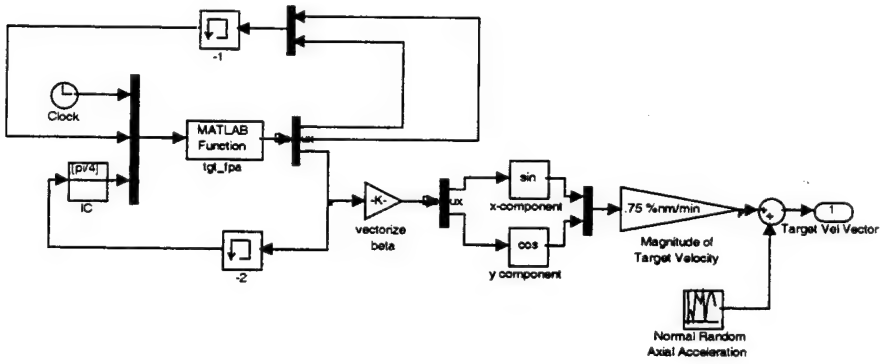
References:

None

M.file File Headers:

```
% function [f] = tgt_chooser(k)
% This function selects the target motion input file from 5 possible inputs. The
% velocity information from the file is extracted into the SIMULINK model to
% provide target movement data. Position data from the file is not used.
```

Target Velocity Vector
for Constrained Random Motion



Brief Description:

When not moving along a road, a target is allowed to move with limited freedom. Direction of target motion can be changed every 9.1 seconds. The target will make a turn of +/- 11.25 degrees 80% of the time. If a turn is made, it will be in the same direction as the last turn 70% of the time. Approximately 2% of the time that a turn is made, it will be a ninety-degree turn to represent evasive travel. Upon completion of a movement, the direction of the velocity vector will remain constant until the next turn opportunity.

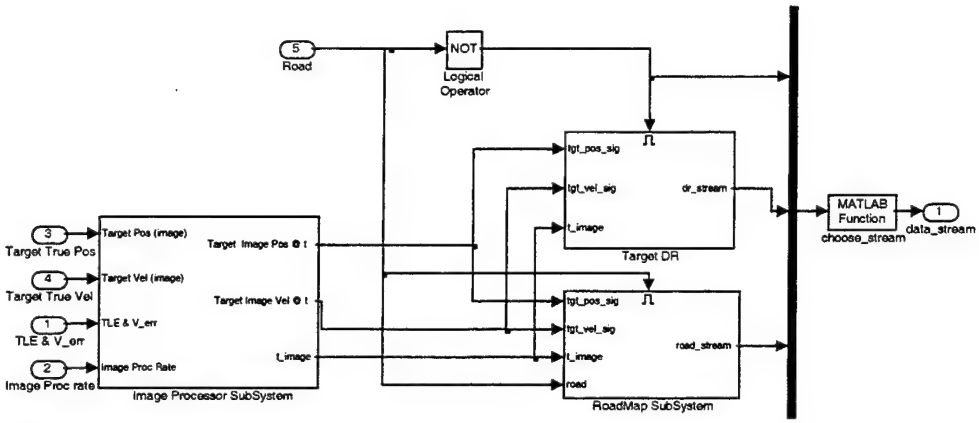
References:

None

M.file File Headers:

```
% function [f] = tgt_truth(k)
% This function selects provides random target motion control when the simulation
% is running in the 'offroad' mode
```

Sensor/Tracker Subsystem



Brief Description:

The "Sensor/Tracker Subsystem" interfaces with the target track via an assumed sensor, and provides predicted target motion data to the "Transmission Subsystem." The "Sensor/Tracker Subsystem" is further divided into an "Image Processor Subsystem", and a choice of either a "RoadMap" or "Target DR" subsystem, which is enabled by the choice of target motion, either road or off-road (constrained random motion).

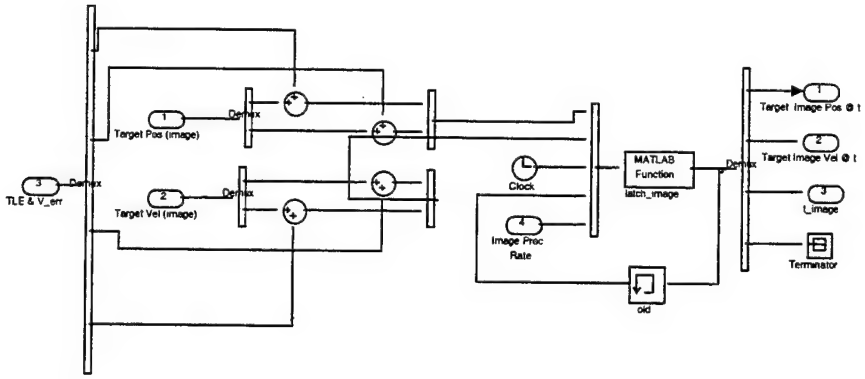
References:

None

M.file File Headers:

```
% function [f] = choose_stream(k)
% This function selects either the data generated from the random target motion
% subsystem or the roadmap subsystem.
```

Image Processor Subsystem



Brief Description:

The "Image Processor Subsystem" represents the hardware and software that interface with the raw imagery received from the sensor. Because a sensor model was not required to complete this research, to represent this activity the target truth position and velocity are input and a uniform random number generator is used to add errors to the raw truth data. This subsystem also incorporates a delay module that holds the now-corrupted data for a nominal time before forwarding it for further processing

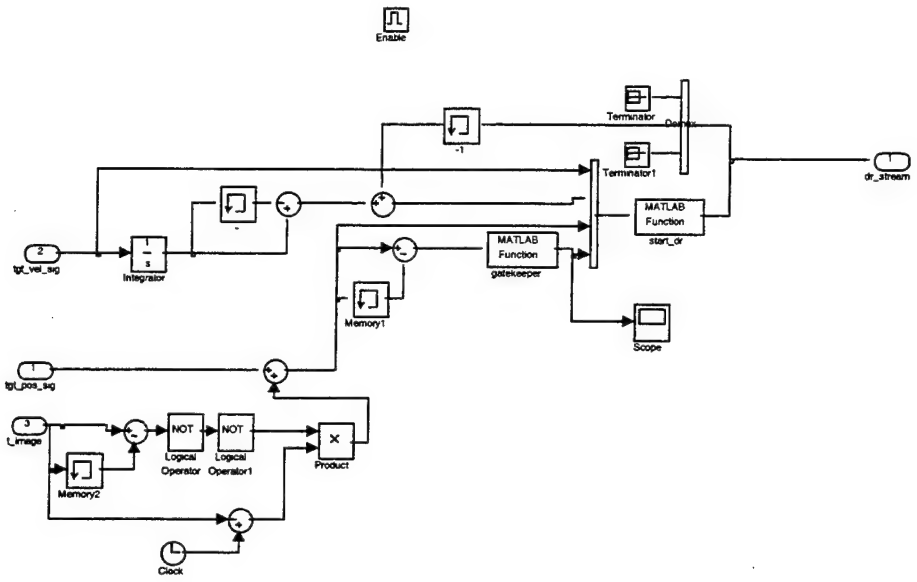
References:

None

M.file File Headers:

```
% function [out] = latch_image(in)
% This function 'latches' the data derived from the target 'image' until the
% next image time arrives based on a nominal fixed interval and a random
% +/- interval.
```

Target DR Subsystem



Brief Description:

The "Target DR Subsystem" is used when the target is not moving along a predictable track or road; the target is free to move in any direction. Due to the expected low utilization of this type of free-motion engagement, the implementation of this part model is extremely simplified, and the model does not attempt to predict future motion.

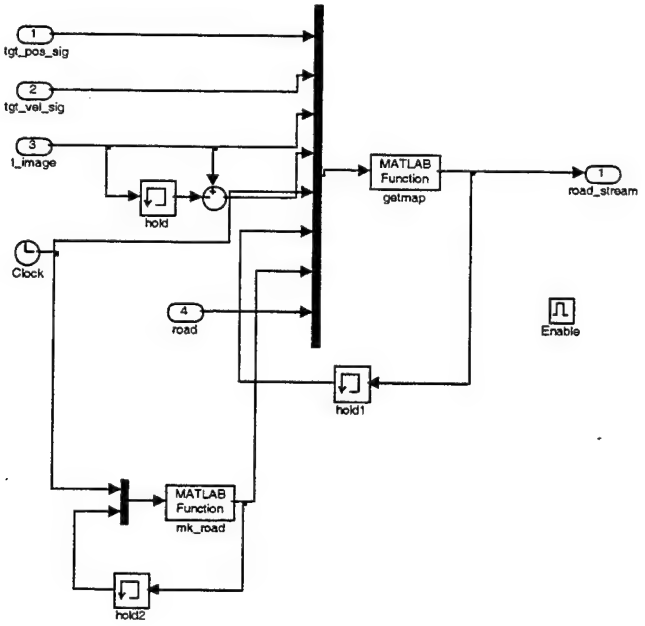
References:

None

M.file File Headers:

```
% function [f] = gatekeeper(k)
% This function controls when a new dr track is initiated based on a change in
% target coordinates provided by the imaging subsystem .

% function [f] = start_dr(k)
% This function starts the DR track in the ground station for a randomly maneuvering
% target.
```

RoadMap Subsystem



Brief Description:

The "Road Map Subsystem" represents a smart system which receives the image "derived" data from the Image Processor, and uses a data base of geographic features to remove some error from the derived data, and predict future location of the target along a road network. The road models in the "Road Map Subsystem" are the same models used in the "Target Truth Subsystem" to develop target motion files, however the "Road Map Subsystem" does not have access to true target motion data (position or velocity).

References:

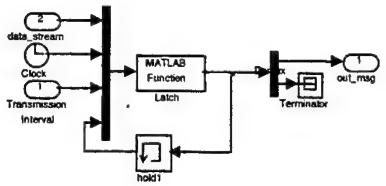
None

M.file File Headers:

```
% function [f] = getmap(k)
% This function takes a vector representation of a road and uses the information
% to predict target motion in the future. The output is a vector data stream
% which contains 60 data sets at 1 second intervals from the time the image was
% developed. The format of the data stream is:
%     f = [(t_image*60+60) px_last py_last vx_last vy_last mat];
% where 'mat' is a set of positions spaced 1 second apart in the form: 't, px, py'.
% The last 3 elements in 'mat' are the same as the first 3 elements in the 'f'
% vector. If the target is not moving on a road, t_max is set to '999' as a flag value,
% and the mat stream is zeros(1,185).
```

```
% function [f] = mk_road(k)
% This function creates a vector representation of a road for use in predicting
% target motion.
```

Transmission Subsystem



Brief Description:

The "Transmission Subsystem" stores the most current data received until it is time for the next transmission. If multiple updates occur between transmissions, only the most current data is transmitted. Transmission is modeled as perfect/instantaneous except that a random dropout rate of 2% is included, which represents the combined effects of attenuation, positioning, and interference that could occur within a mission.

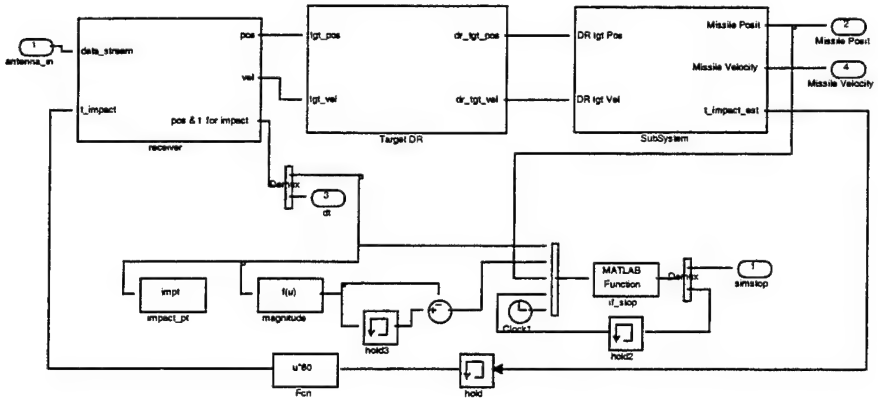
References:

None

M.file File Headers:

```
% function [f] = latch(k)
% This function holds the last data entered into the datastream and transmits
% every 2 seconds.
```

Missile Behavior Subsystem



Brief Description:

The "Missile Behavior Subsystem" represents an idealized weapon assigned to attack a single target. Kinematics of a generic weapon are represented by computing a lead navigation solution and tracking the convergence of the computed position along that navigation path and the predicted location of the target from the "Sensor/Tracker Subsystem." The weapon does not transmit its own position or predicted intercept information to any other part of the system.

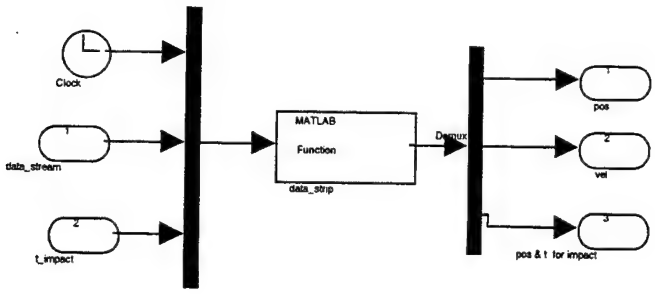
References:

The algorithm used was adapted from Zarchan, *Tactical and Strategic Missile Guidance, Second Edition*, chapter 2, listing 2.1.

M.file File Headers:

```
% function [f] = if_stop(k)
% This function stops the simulation if the difference between the last estimated
% impact position and the missile position increase after it has decreased below
% 1/10 mile. Note that in a 'real' missile, it would have guided to impact at the impact point,
% flyby could not actually occur. In case the missile is a 'miss' and does not enter
% the 1/10 mile radius, the simulation will stop at 4 minutes and 54 seconds since
% there is only 5 minutes worth of target motion data stored in the data files.
```

Missile Receiver Subsystem



Brief Description:

The Missile Receiver Subsystem receives the data from the Data Transmission Subsystem and converts it to a form usable by the missile.

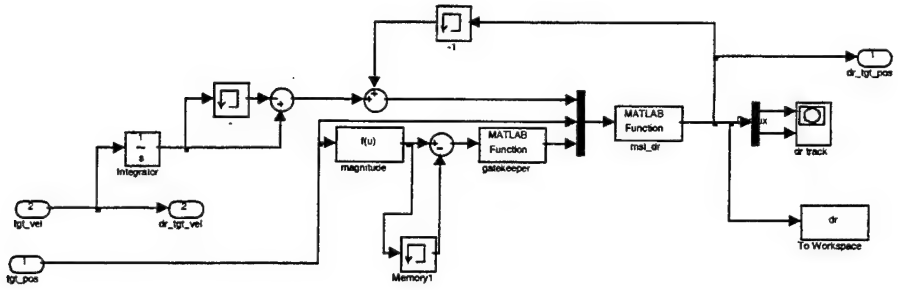
References:

None

M.file File Headers:

```
% function [f] = data_strip(k)
% This function receives the data stream transmission and parses the data, derives
% the necessary values, and passes them to the missile for calculation and behavior.
% The basic algorithm computes a predicted impact point based on target estimated
% position and velocity, and creates a pseudo-position linearly spaced back from the
% impact position using velocity and time to impact to start the dr track for
% missile guidance.
```

Missile DR Subsystem



Brief Description:

The Missile DR Subsystem accepts the updated target information from the receiver and initiates a DR track for the current target. Each time new information is available from the receiver, a new DR track is initiated.

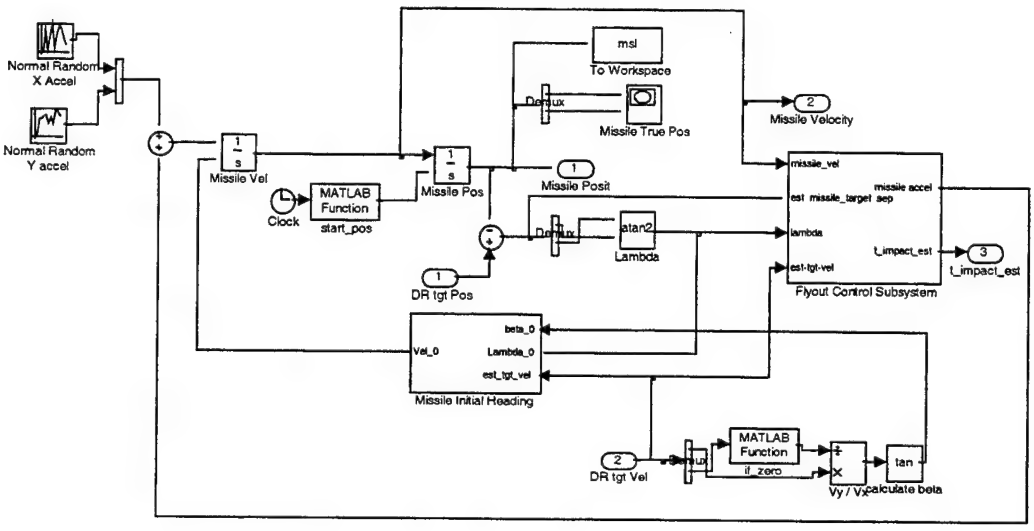
References:

None

M.file File Headers:

```
% function [f] = gatekeeper(k)
% This function controls when a new dr track is initiated based on a change in
% target coordinates provided by the imaging subsystem

% function [f] = msl_dr(k)
% This function chooses either the existing dr track, or updated position data
% based on when a new dr track is initiated based on a change in position data
% in the stream.  The function is controlled by gatekeeper.m
```

Missile Navigation Subsystem



Brief Description:

The Navigation Subsystem controls missile motion based on target position at estimated time of intercept, time of flight remaining, and calculated lead angle.

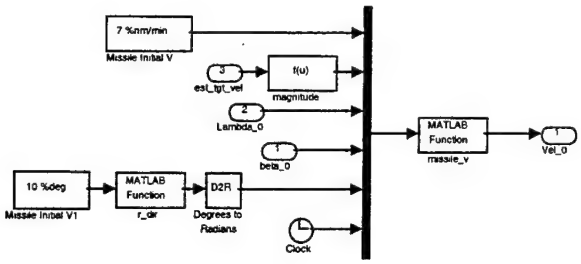
References:

The algorithm used was adapted from Zarchan, *Tactical and Strategic Missile Guidance, Second Edition*, chapter 2, listing 2.1.

M.file File Headers:

```
% function [f] = if_zero(k)
% This function prevents a 'divide by zero' error

% function [f] = start_pos(k)
% This function initializes the position of the missile at the start of the simulation
```

Missile Initial Heading Subsystem



Brief Description:

This subsystem initializes the missile heading at simulation start (missile launch) and includes a maximum 10 degree heading error.

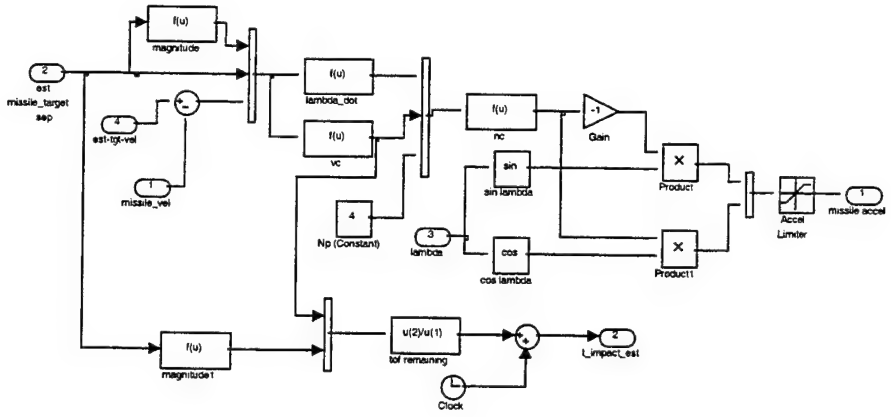
References:

None

M.file File Headers:

```
% function [f] = missile_v(k)
% This function computes the initial missile heading vector

% function [f] = r_dir(k)
% This function provides a random direction for the missile initial heading error
```

Missile Flyout Control Subsystem



Brief Description:

This subsystem controls missile heading updates during the engagement.

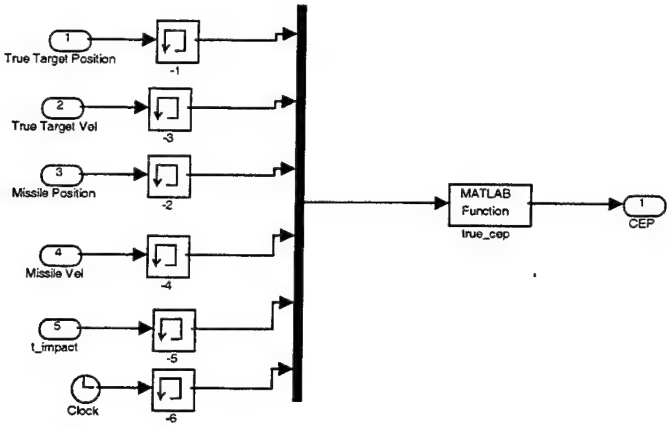
References:

The algorithm used was adapted from Zarchan, *Tactical and Strategic Missile Guidance, Second Edition*, chapter 2, listing 2.1.

M.file File Headers:

None

Display Subsystem



Brief Description:

The "Display Subsystem" accepts position and velocity data from the "Target Truth" and "Missile Behavior" subsystems, and calculates the radial miss distance from the missile location at time of impact to the true target position.

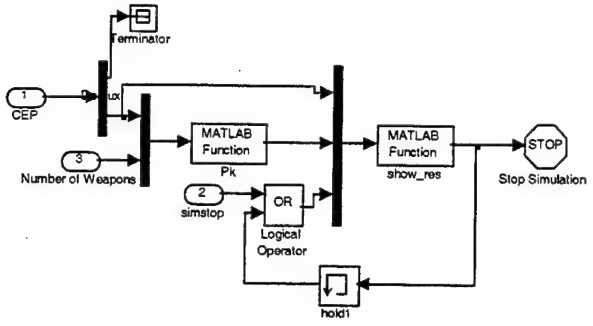
References:

None

M.file File Headers:

```
% function [f] = true_cep(k)
% This function calculates the radial miss distance (rmd) from the missile
% to the true target position at the time of weapon 'impact'.
```

Lethality Subsystem



Brief Description:

This subsystem receives the radial miss distance from the "Display Subsystem" and calculates an estimated single sortie probability of damage (SSPD) using generic target dimensions and warhead measures of effectiveness. This subsystem also issues the "stop simulation" command.

References:

None

M.file File Headers:

```
% function [SSPD] = Pk(k)
% This function calculates the single shot probability of damage (SSPD) using the
% radial miss distance (rmd) from the missile to the true target position at the
% time of weapon 'impact'. Values used are generic to be unclassified.

% function [st] = show_res(k)
% This function displays the miss distance and the the single shot probability
% of damage (SSPD), and generates the simulation stop value.
% Reuslts are derived using generic values and are unclassified.
```

APPENDIX C. MATLAB® M-FILES

FILE Name: choose_stream.m

```
function[f] = choose_stream(k)

% function [f] = choose_stream(k)
% This function selects either the data generated from the random target motion
% subsystem or the roadmap subsystem.

% This function supports MATLAB SIMULINK model "tcmts.mdl" using MATLAB
% version 5.3.0.10183 (R11).
% This was developed by CDR R. L. Mahr for the Naval Postgraduate School in partial
% fulfillment of the requirements for a Masters Degree in Aeronautical Engineering
% January 2001.  All rights reserved.

choice = k(1);          % inverted road selection
random = k(2:191);      % data from random motion subsystem
road = k(192:381);      % data from roadmap subsystem

if choice == 1
    f = random;
else
    f = road;
end % if choice

% end file
```

FILE Name: data_strip.m

```
function [f] = data_strip(k)

% function [f] = data_strip(k)
% This function receives the data stream transmission and parses the data, derives
% the necessary values, and passes them to the missile for calculation and behavior.
% The basic algorithm computes a predicted impact point based on target estimated
% position and velocity, and creates a pseudo-position linearly spaced back from the
% impact position using velocity and time to impact to start the dr track for
% missile guidance.

% This function supports MATLAB SIMULINK model "tcmts.mdl" using MATLAB
% version 5.3.0.10183 (R11).
% This was developed by CDR R. L. Mahr for the Naval Postgraduate School in partial
% fulfillment of the requirements for a Masters Degree in Aeronautical Engineering
% January 2001.  All rights reserved.

t_now = k(1)*60;          % time now, s
t_max = k(2);             % time of maximum data prediction values in data stream
fcast_pos = k(3:4);       % target position at maximum data stream time
fcast_vel = k(5:6);       % target velocity at maximum data stream time
stream = k(7:191);        % data stream in format time, pos_x, pos_y
t_impact = k(192);        % calculated impact time from missile

% set initial target data to first stream datapoint, which is the target position
% at the time the image was obtained.  Derive target velocity using the next position
% which is 1 second later (by design of the data stream).

px = stream(2);                  % target position at time of image, NM
py = stream(3);                  % target position at time of image, NM
vx = (stream(5)-stream(2))*60;   % target velocity, NM/min
vy = (stream(6)-stream(3))*60;   % target velocity, NM/min

% from data stream, strip out the target position at predicted impact time
% or if a non-predictable target, using straight vector prediction

% '999' is flag to indicate non-road based (fully random) target
if t_max == 999          % '999' is flag to indicate non-road based (fully random) target
    pf = fcast_pos+(t_impact-t_now)*(fcast_vel/60);  % predict position at t_impact
    xi = pf(1);
    yi = pf(2);
    px = fcast_pos(1);   %assign current pos and velocity
    py = fcast_pos(2);
    vx = fcast_vel(1);
    vy = fcast_vel(2);
% if impact time is greater than 1-sec prediction stream, compute predicted impact point
% based on forecast pos and velocity at max time provided
elseif t_impact >= t_max
    pf = fcast_pos+(t_impact-t_max)*fcast_vel;   % predict position at t_impact
    xi = pf(1);
    yi = pf(2);
    alfa = atan2(fcast_vel(2),fcast_vel(1));
    v_mag = sqrt(fcast_vel(1)^2+fcast_vel(2)^2);
```

```
    px = pf(1) - (t_impact-t_now)*v_mag*cos(alfa); % create pseudo position for dr track
    py = pf(2) - (t_impact-t_now)*v_mag*sin(alfa);
    vx = v_mag*cos(alfa)*60;
    vy = v_mag*sin(alfa)*60;
% if impact time is less than maximum data time, find closest-next data time.  Note
% data times are every third position in the data stream.
else
    for i = 4:3:183
        if t_impact <= stream(i)
            dt = stream(i) - t_impact;          % impact time within the 1 sec data interval
            vx = stream(i+1) - stream(i-2);
            vy = stream(i+2) - stream(i-1);
            xi = stream(i+1) - dt*vx;           % predict impact position using dt
            yi = stream(i+2) - dt*vy;
            adv_t = stream(i)-t_now;            % create pseudo position for dr track
            px = stream(i+1) - vx*adv_t;
            py = stream(i+2) - vy*adv_t;
            vx = vx*60;
            vy = vy*60;
            break % for i
        end % if t_imapct
    end %for i
end % if t_max

f = [px py vx vy xi yi t_impact];

% end file
```

FILE Name: gatekeeper.m

```
function [lock] = gatekeeper(k)

% function [f] = gatekeeper(k)
% This function controls when a new dr track is initiated based on a change in
% target coordinates provided by the imaging subsystem

% This function supports MATLAB SIMULINK model "tcmts.mdl" using MATLAB
% version 5.3.0.10183 (R11).
% This was developed by CDR R. L. Mahr for the Naval Postgraduate School in partial
% fulfillment of the requirements for a Masters Degree in Aeronautical Engineering
% January 2001.  All rights reserved.

test = k;      % result of subtracting the last target position from the current

if test == 0
    lock = 0;
else
    lock = 1;
end % if test

% end file
```

FILE Name: getmap.m

```
function [f] = getmap(k)

% function [f] = getmap(k)
% This function takes a vector representation of a road and uses the information
% to predict target motion in the future. The output is a vector data stream
% which contains 60 data sets at 1 second intervals from the time the image was
% developed. The format of the data stream is:
%     f = [(t_image*60+60) px_last py_last vx_last vy_last mat];
% where 'mat' is a set of positions spaced 1 second apart in the form: 't, px, py'.
% The last 3 elements in 'mat' are the same as the first 3 elements in the 'f'
% vector. If the target is not moving on a road, t_max is set to '999' as a flag value,
% and the mat stream is zeros(1,185).

% This function supports MATLAB SIMULINK model "tcmts.mdl" using MATLAB
% version 5.3.0.10183 (R11).
% This was developed by CDR R. L. Mahr for the Naval Postgraduate School in partial
% fulfillment of the requirements for a Masters Degree in Aeronautical Engineering
% January 2001. All rights reserved.

tgt_pos = k(1:2);          % target position estimate at time of image
tgt_vel = k(3:4)/60;       % target velocity estimate at time of image
t_image = k(5);            % time of image
new_im = k(6);             % variable to indicate new image data (0 = old image)
t_now = k(7);              % time now
old_data = k(8:197);       % old data stream
road_map = k(198:835);% vector data representing road
road_choice = k(836);   % choice of road (1 or 2 = straight, 3 or 4 = turning)

% load datastream at start of simulation
if t_now <.002
    new_im = 1;
end

% if new image data is available, construct new data stream
if new_im ~= 0

% IMPORT ROAD VECTOR DATA
% choice of road (1 or 2 = straight, 3 or 4 = turning)
    if road_choice == 1 | road_choice == 2
        route = [0 0; 10 10; 20 20; 100 100];
    else
        kip = length(road_map)-1;
        route = [road_map(1) road_map(2)];
        for n = 3:2:kip
            route = [route; road_map(n) road_map(n+1)];
        end % for
    end % if road_choice
% END IMPORT ROAD VECTOR DATA

    points = length(route)-1; % don't read past of last road turnpoint
```

```
% set initial target position to position at time of image
    px(1) = tgt_pos(1);
    py(1) = tgt_pos(2);

% calculate magnitude of displacement of target from the origin to
% determine position on the road.  Note that this is a simplification
% for this simulation and the simulation roads are always monotonically
% increasing in distance from the origin.
    pos_mag_initial = sqrt(px(1)^2+py(1)^2);

% set initial target velocity to velocity at time of image
    vx = tgt_vel(1);
    vy = tgt_vel(2);

% assume that the velocity magnitude will have less error than the actual
% velocity vectorial direction.  Determine the velocity magnitude and later
% apply it to the vector direction of the road segment to minimize errors
    v_mag = (sqrt(vx^2+vy^2));

% initialize the output matrix datastream
    mat = [t_image*60 px(1) py(1)];

% using the road vector representation, find the next turn IN FRONT of the target
    for n = 2:points
        turn_mag = sqrt(route(n,1)^2+route(n,2)^2);
        if turn_mag > pos_mag_initial
            tn = n;
            break
        end %if turn_mag
    end %for n

% with the next turn IN FRONT of the target identified, loop to create 60 predicted
% movement points spaced 1 second apart.  Note that the algorithm assumes constant
% velocity over the entire timeframe.

    for i = 1:60

% for the present position to the next turnpoint, calculate the distance
        dx_dr = route(tn,1)-px(i);
        dy_dr = route(tn,2)-py(i);
        alfa1 = atan2(dy_dr,dx_dr);          % angle to next turnpoint
        dist2turn = sqrt(dx_dr^2+dy_dr^2);

% calculate the angle of the road to the next turnpoint
        dx_route = route(tn,1)-route(tn-1,1);
        dy_route = route(tn,2)-route(tn-1,2);

% assuming the velocity derived from the image contains errors in direction, take
% the magnitude of the vector and assume only that is correct
        dist_trav = v_mag;

% for short vector segments the distance traveled in one 'move' may be greater than
% the distance to the next turn.  For each move, subtract off the distance to the
% next turn until the move ends short of a turnpoint.
        while dist_trav >= dist2turn
```

```
            dist_trav = dist_trav - dist2turn;
            dx_route = route(tn+1,1)-route(tn,1);
            dy_route = route(tn+1,2)-route(tn,2);
            dist2turn = sqrt(dx_route^2+dy_route^2);
            tn = tn+1;
        end %while

% calculate the angle of the road between the turnpoint 'behind' (th-1) and the
% turnpoint 'in front' of the target (tn).
        alfa = atan2(dy_route, dx_route);

% test - if the distance traveled is less than the magnitude of the velocity vector,
% at least one turn must have been made during the move, so add the movement vector
% to the position of the turnpoint 'behind' the target.  If distance traveled equals
% the magnitude of the velocity vector, no turns have been made, so subtract movement
% from next turn position.  Note: adding it to the last target position does not
% eliminate crosstrack error on long legs.
        if dist_trav < v_mag
            rem = dist_trav*[cos(alfa) sin(alfa)];      % distance to move (remaining)
            px(i+1) = route(tn-1,1)+rem(1);
            py(i+1) = route(tn-1,2)+rem(2);
        else
            mov_x = cos(alfa)*cos(alfa-alfa1)*(dist2turn-v_mag);
            mov_y = sin(alfa)*cos(alfa-alfa1)*(dist2turn-v_mag);
            px(i+1) = route(tn,1) - mov_x;
            py(i+1) = route(tn,2) - mov_y;
        end %if

% add next point to data stream vector
        t_next = (t_image*60)+(i);
        mat = [mat t_next px(i+1) py(i+1)];

    end %for

% data stream must be constant length (190 elements)
    dim = size(mat);
    if dim(2) < 185
        xcs = 185-dim(2);
        z = zeros(1,xcs);
        mat = [mat z];
    end

% output vector is width 190
    px_last = px(i);
    py_last = py(i);
    vx_last = v_mag*cos(alfa);
    vy_last = v_mag*sin(alfa);

    f = [(t_image*60+60) px_last py_last vx_last vy_last mat];
```

```
% PLOT 'THE CATEPILLAR'
% plot(route(:,1),route(:,2), px, py, '*:') % this plots the 'caterpillar'

% if there is no new image, repeat old data
else

    f = [old_data];

end % if new_image

% end file
```

FILE Name: if_stop.m

```
function [f] = if_stop(k)

% function [f] = if_stop(k)
% This function stops the simulation if the difference between the last estimated
% impact position and the missile position increase after it has decreased below
% 1/10 mile. Note that in a 'real' missile, it would have guided to impact at the impact point,
% flyby could not actually occur. In case the missile is a 'miss' and does not enter
% the 1/10 mile radius, the simulation will stop at 4 minutes and 54 seconds since
% there is only 5 minutes worth of target motion data stored in the data files.

% This function supports MATLAB SIMULINK model "tcmts.mdl" using MATLAB
% version 5.3.0.10183 (R11).
% This was developed by CDR R. L. Mahr for the Naval Postgraduate School in partial
% fulfillment of the requirements for a Masters Degree in Aeronautical Engineering
% January 2001. All rights reserved.

xi = k(1);          % x impact point
yi = k(2);          % y impact point
steady = k(3);      % relative measure of 'steadiness' of impact point
xm = k(4);          % x position of missile
ym = k(5);          % y position of missile
last_sep = k(6);    % stored value of last separation
t_elapsed = k(7);   % elapsed time since start of sim

% calculate rms separation
sep = sqrt((xi-xm)^2+(yi-ym)^2);

% stop sim if this iteration has more separation than last (assumes missile is past impact
% point. Note that in a 'real' missile, it would have guided to impact at the impact point,
% flyby could not actually occur.

if sep >= last_sep
    if sep <= .1 & steady <= .0011
        stop = 1;
    else
        last_sep = sep;
        stop = 0;
    end %if sep <=
else
    last_sep = sep;
    stop = 0;
end % if sep >=

% stop simulation if near end of target motion data in file (5 mins)
if t_elapsed > 4.9
    stop = 1;
end % if t_elapsed

f = [stop last_sep];

% end file
```

FILE Name: if_zero.m

```
function f = if_zero(k)

% function [f] = if_zero(k)
% This function prevents a 'divide by zero' error

% This function supports MATLAB SIMULINK model "tcmts.mdl" using MATLAB
% version 5.3.0.10183 (R11).
% This was developed by CDR R. L. Mahr for the Naval Postgraduate School in partial
% fulfillment of the requirements for a Masters Degree in Aeronautical Engineering
% January 2001.  All rights reserved.

var = k;

if var <= 10^-15

    var = 10^-15;

end % if var

f = var;

% end file
```

FILE Name: latch.m

```
function [out] = latch(in)

% function [f] = latch(k)
% This function holds the last data entered into the datastream and transmits
% every 2 seconds.

% This function supports MATLAB SIMULINK model "tcmts.mdl" using MATLAB
% version 5.3.0.10183 (R11).
% This was developed by CDR R. L. Mahr for the Naval Postgraduate School in partial
% fulfillment of the requirements for a Masters Degree in Aeronautical Engineering
% January 2001.  All rights reserved.

x = in(1:190);
t = in(191);
rate = in(192);
old = in(193:382);
t_next = in(383);

if t<=.005
    t_next = t+(rate);
    out = [x' t_next];
end

if t >= t_next
    skip = rand;
        if skip >=.98
            rate = rate*2;
        end
    t_next = t+(rate);
    out = [x' t_next];
else
    out = [old' t_next];
end

% end file
```

FILE Name: latch_image.m

```
function [out] = latch_image(in)

% function [out] = latch_image(in)
% This function 'latches' the data derived from the target 'image' until the
% next image time arrives based on a nominal fixed interval and a random
% +/- interval.

% This function supports MATLAB SIMULINK model "tcmts.mdl" using MATLAB
% version 5.3.0.10183 (R11).
% This was developed by CDR R. L. Mahr for the Naval Postgraduate School in partial
% fulfillment of the requirements for a Masters Degree in Aeronautical Engineering
% January 2001.  All rights reserved.

new_info = in(1:4);      % data based on current true position and velocity + TLE & VE
t_now = in(5);           % time now
last_trans = in(6:10);   % latched data from last transmission
t_next = in(11);         % time of next transmission
interval = in(12);       % fixed nominal interval between transmissions

if interval > (1/60)
    r_proc = randn/10;
else
    r_proc = 0;
end

% load data stream with new info at beginning of sim
if t_now <=.005
    tgt_pv = new_info;                  % forward new data
    t_image = t_now;                    % forward time of image
    t_next = t_now+interval+r_proc; % create next image time
    out = [tgt_pv' t_image t_next];
end % if t-now

% at next transmit time
if t_now >= t_next
    tgt_pv = new_info;                  % forward new data
    t_image = t_now;                    % forward time of image
    t_next = t_now+interval+r_proc; % create next image time
    out = [tgt_pv' t_image t_next];
else
    out = [last_trans' t_next];
end % if t_now

% end file
```

FILE Name: missile_v.m

```
function [f] = missile_v(k)

% function [f] = missile_v(k)
% This function computes the initial missile heading vector

% This function supports MATLAB SIMULINK model "tcmts.mdl" using MATLAB
% version 5.3.0.10183 (R11).
% This was developed by CDR R. L. Mahr for the Naval Postgraduate School in partial
% fulfillment of the requirements for a Masters Degree in Aeronautical Engineering
% January 2001.  All rights reserved.

vm_0 = k(1);        % magnitude of missile velocity, NM/min
vt_0 = k(2);        % estimated target velocity, NM/min
lambda = k(3);      % target velocity vector angle from 'horizontal'
beta = k(4);        % missile velocity vector angle from 'horizontal'
HE = k(5);          % initial heading error at launch
t = k(6);           % time now

% at t = 0, compute missile initial heading

if t==0
    lead = asin(vt_0*sin(beta+lambda)/vm_0);                    % lead angle
    vm = [vm_0*cos(lead+lambda+HE);vm_0*sin(lead+lambda+HE)];   % velocity vector
    f = [vm];
else
    f = [0;0];
end % end if t

% end file
```

FILE Name: mk_road.m

```
function [f] = mk_road(k)

% function [f] = mk_road(k)
% This function creates a vector representation of a road for use in predicting
% target motion.

% This function supports MATLAB SIMULINK model "tcmts.mdl" using MATLAB
% version 5.3.0.10183 (R11).
% This was developed by CDR R. L. Mahr for the Naval Postgraduate School in partial
% fulfillment of the requirements for a Masters Degree in Aeronautical Engineering
% January 2001.  All rights reserved.

t = k(1);                    % time now
road_vector = k(2:639); % static road vector

% the creation of the road significantly slows down the simulation, so it
% is created only for the first 2 iterations, at which point the memory loop
% is full, and then data can be treated as a vector.
if t < 0.002

% CONSTRUCT ROAD
    route = [0 0 0 1];          % starting point and first turn point

% create a radiused turn
    x0 = 0.5;
    y0 = 1;
    radius = 0.5;
    i = 1;

    for z = 0:.005:pi/2
        x(i) = x0+radius*sin(z);
        y(i) = y0+radius*(1-cos(z));
        route = [route  x(i) y(i)];
        i = i+1;
    end

    route = [route  5 5  10 10];        % last two turn points
% END CONSTRUCT ROAD

    f = route;
else

    f = road_vector;

end % if t

% end file
```

FILE Name: msl_dr.m

```
function [f] = msl_dr(k)

% function [f] = msl_dr(k)
% This function chooses either the existing dr track, or updated position data
% based on when a new dr track is initiated based on a change in position data
% in the stream.  The function is controlled by gatekeeper.m

% This function supports MATLAB SIMULINK model "tcmts.mdl" using MATLAB
% version 5.3.0.10183 (R11).
% This was developed by CDR R. L. Mahr for the Naval Postgraduate School in partial
% fulfillment of the requirements for a Masters Degree in Aeronautical Engineering
% January 2001.  All rights reserved.

dr_pos = k(1:2);          % current dr track position estimate
new_pos = k(3:4);         % updated position coordinates
unlock = k(5);            % control command from gatekeeper.m

if unlock == 1            % if new position, gatekeeper will send 1
    f = new_pos;
else
    f = dr_pos;
end %if unlock

% end file
```

FILE Name: Pk.m

```
function [SSPD] = Pk(k)

% function [SSPD] = Pk(k)
% This function calculates the single shot probability of damage (SSPD) using the
% radial miss distance (rmd) from the missile to the true target position at the
% time of weapon 'impact'.  Values used are generic to be unclassified.

% This function supports MATLAB SIMULINK model "tcmts.mdl" using MATLAB
% version 5.3.0.10183 (R11).
% This was developed by CDR R. L. Mahr for the Naval Postgraduate School in partial
% fulfillment of the requirements for a Masters Degree in Aeronautical Engineering
% January 2001.  All rights reserved.

r_wpn = 0.95;        % reliability of the weapon after launch

maef = 10000;        % mean area of effective frag
p_hit = .3;          % probability of hit
p_nm = .640;         % probability of near miss
Pd_hit = 0.154;      % probability of damage given a hit

tgt_len = 30;        % generic target length
tgt_width = 10;
lw_ratio = tgt_width/tgt_len;

let = 1.128*sqrt(maef*lw_ratio);
wet = let/lw_ratio;

cep = k(1);

rep = cep*0.573;
dep = rep;

sspd_r = let/(sqrt((17.6*rep^2)+let^2));
sspd_d = wet/(sqrt((17.6*rep^2)+wet^2));

sspd1 = sspd_r * sspd_d;
sspd2 = 0;

sspd_one = r_wpn * Pd_hit * (sspd1*p_nm + sspd2*p_hit);

num_wpns = k(2);

SSPD = 1 - (1-sspd_one)^num_wpns;

% end file
```

FILE Name: r_dir.m

```
function [f] = r_dir(k)

% function [f] = r_dir(k)
% This function provides a random direction for the missile initial heading error

% This function supports MATLAB SIMULINK model "tcmts.mdl" using MATLAB
% version 5.3.0.10183 (R11).
% This was developed by CDR R. L. Mahr for the Naval Postgraduate School in partial
% fulfillment of the requirements for a Masters Degree in Aeronautical Engineering
% January 2001.  All rights reserved.


HE = k;       % magnitude of the initial heading error

u = 1;

% set direction sign based on random number
if rand>0.5
    u = -u;
end % if rand

f = u*HE*rand;

% end file
```

FILE Name: show_res.m

```
function [st] = show_res(k)

% function [st] = show_res(k)
% This function displays the miss distance and the  single shot probability
% of damage (SSPD), and generates the simulation stop value.
% Results are derived using generic values and are unclassified.

% This function supports MATLAB SIMULINK model "tcmts.mdl" using MATLAB
% version 5.3.0.10183 (R11).
% This was developed by CDR R. L. Mahr for the Naval Postgraduate School in partial
% fulfillment of the requirements for a Masters Degree in Aeronautical Engineering
% January 2001.  All rights reserved.

miss_dist = k(1);          % radial miss distance
sspd = k(2);               % sspd
simstop = k(3);            % simulation stop input 0 = continue, 1 = stop

% if criteria have been met to stop the simulation, display results
if simstop > 0
    dist = num2str(miss_dist);
    SSPD = num2str(sspd);
    st = 1;                % if simstop commanded, set stop value to 1
    F1 = ['END:   CEP (ft) = ' dist '   SSPD = ' SSPD];
    disp(F1)
    break
else
    st = 0;                % if simstop not commanded, continue
end


% end file
```

FILE Name: start_dr.m

```
function [f] = start_dr(k)

% function [f] = start_dr(k)
% This function starts the DR track in the ground station for a randomly maneuvering
% target.

% This function supports MATLAB SIMULINK model "tcmts.mdl" using MATLAB
% version 5.3.0.10183 (R11).
% This was developed by CDR R. L. Mahr for the Naval Postgraduate School in partial
% fulfillment of the requirements for a Masters Degree in Aeronautical Engineering
% January 2001.  All rights reserved.

dr_vel = k(1:2);          % dr track velocity
dr_pos = k(3:4);          % dr track position
new_pos = k(5:6);         % new position from imaging system
unlock = k(7);            % unlock command from 'gatekeeper.m'

% '999' is a flag to the missile to indicate the track is based on random target and
% no position predictions are provided
if unlock == 1
    g = [999 new_pos' dr_vel'];
else
    g = [999 dr_pos' dr_vel'];
end

z = zeros(1,185);

f = [g z];

% end file
```

FILE Name: start_pos.m

```
function [f] = start_pos(k)

% function [f] = start_pos(k)
% This function initializes the position of the missile at the start of the simulation

% This function supports MATLAB SIMULINK model "tcmts.mdl" using MATLAB
% version 5.3.0.10183 (R11).
% This was developed by CDR R. L. Mahr for the Naval Postgraduate School in partial
% fulfillment of the requirements for a Masters Degree in Aeronautical Engineering
% January 2001.  All rights reserved.

t = k;          % simulation time

% if simulation time is 0, initialize missile position
if t == 0

% x position not more than 20 NM from target (0,0)
    x = rand;
    if x>0.5
        u = -1;
    else
        u = 1;
    end % if rand
    xpos = u*(20*rand); %missile initial x-offset


% y position not more than 20 NM from target (0,0)
    y = rand;
    if y>0.5
        u = -1;
    else
        u = 1;
    end % if rand
    ypos = u*(20*rand); %missile initial y-offset

% missile must initialize at least 5 NM from target
    if sqrt(xpos^2+ypos^2) < 5
        xy = rand;
        if xy>0.5
            if sign(ypos) == -1
                ypos = ypos - 5;
            else
                ypos = ypos + 5;
            end % if sign
        else
            if sign(xpos) == -1
                xpos = xpos - 5;
            else
                xpos = xpos + 5;
            end %if sign
        end % if xy
    end % if sqrt
```

```
f = [xpos ypos];
else
f = [100 100];
end % if t
% end file
```

FILE Name: test.m

```
% file test.m
%
% This file executes multiple sequential runs of the "tcmts.mdl" and changes the inputs
% for each run to user specified values.  To implement this file, the "Inputs and
% Errors Subsystem" block values should be set to the variable names specified below.
% For the research, a copy of the model was made which was called "tcmts_multi.mdl",
% which incorporated the variable names, and left the original model to be used for
% single run tests.  The model must also contain a "to workspace" block in the
% Lethality Subsystem which takes the [RE DE CEP] input from the 'CEP' input, and
% mux together with the simulation stop output of the "show_res" block to provide a (4,1)
% output to the workspace variable "report" [RE DE CEP Simstop].
%
% This function supports MATLAB SIMULINK model "tcmts.mdl" using MATLAB
% version 5.3.0.10183 (R11).
% This was developed by CDR R. L. Mahr for the Naval Postgraduate School in partial
% fulfillment of the requirements for a Masters Degree in Aeronautical Engineering
% January 2001.  All rights reserved.

% SET VARIABLES AND SEEDS TO DEFAULT VALUES
%
% A & B are random number generator seeds for TLE
A = 1;
B = 3;
% C & D are random number generator seeds for VE
C = 2;
D = 4;
% E is choice of road (0-4)
E = 4;
% F is image processing interval in form X/60
%       nominal value F = 20/60
F = 20/60;
% G is data transmission interval in form X/60
%       nominal value G = 2/60
G = 2/60;
% H is TLE/VE on/off switch (1 = on, 0 = off)
H = 1;
% I is maximum size of TLE in form X/6075
%       nominal value I = 50/6075
I = 50/6075;
% J is TLE only on/off switch (1 = on, 0 = off)
J = 1;
% K is maximum size of VE in form X/60
%       nominal value K = 3/60
K = 3/60;
% L is VE only on/off switch (1 = on, 0 = off)
L = 1;

% EXECUTE DATA RUNS (OUTER LOOP)

for w = 1:2             % w is number of variables to be run.  This will
                        % be the number of times the outer loop executes
```

```
if w == 1
    F = 10/60;        % Set variable to be modified on the first run
else
    F = 20/60;        % Set variable to be modified on the second run
end %if w

CEP = [ 0 ];          % initialize CEP matrix
thesis = [1];         % initialize thesis matrix

for k = 2:105                   % there will be 100 iterations of the inner loop, plus
                                % 5 derived values appended to the end of the matrix.
                                % To run >100, change to 5 + value of 'i' below.
    thesis = [thesis;k];        % this provides a 'counter' in the first column of 'thesis'
                                % for easy reading of data
end %for k

% EXECUTE ONE VARIABLE RUN SET (INNER LOOP)

for j = 1:4             % for 4 road cases
    E = j;              % set E (road choice) to j

    for i = 1:100           % i is the number of times the model will be iterated.
                            % as set up, this must be '5' less than k above.
        run = [w j i]       % display run sequence to command window to monitor progress
        A = round(10*rand); % initialize seed values for each run
        B = round(10*rand);
        C = round(10*rand);
        D = round(10*rand);

        sim('tcmts_multi')    % call Simulink model with set variables

        for n = 1:10000       % set a large number to exceed number of potential datapoints
            if report(n,4) ~= 0    % checks the status of the stop simulation command
                                   % when stop sim is 'true', report(n,4) = 1.
                if i == 1          % for first loop, initialize the analysis vectors
                    CEP = [report(n,3)]; % this is a vector of radial miss distance
                    RE = [report(n,1)];  % this is a vector of range errors
                    DE = [report(n,2)];  % this is a vector of deflection errors
                else               % on subsequent loops, append new data to existing vector
                    CEP = [CEP; report(n,3)];
                    RE = [RE; report(n,1)];
                    DE = [DE; report(n,2)];
                end %if i
                break   % if data has been stored, break 'for n'
            end %if report
        end % for n
    end  %for i

    % after i runs are complete, sort data in each vector
    CEP = sort(CEP);
    RE = sort(RE);
    DE = sort(DE);
```

```
            % calculate basic statistical parameters for each vector using
            % all available data, including Type-1 gross errors.
            mu = mean(CEP);   % mean of all radial miss distances
            sigma = std(CEP);   % standard deviation of all radial miss distances
            fifty = CEP(50); % CEP of sample
            sigmaR = std(RE);   % standard deviation of range errors
            sigmaD = std(DE);   % standard deviation of deflection errors

            % append statistical data to CEP vector
            CEP = [CEP;fifty;mu;sigma;sigmaR;sigmaD];

            % NOTE:   RE & DE vectors are discarded.  If a statistical
            %         analysis of the underlying distributions is needed,
            %         they can    be saved to the workspace.

            thesis = [thesis CEP];   % append the CEP vector as the next column
                                     % in the thesis matrix.  After all 'j' loops
                                     % the thesis matrix will have j+1 columns
                                     % where the first column is the counter,
                                     % and the last 5 items in each column are the
                                     % statistical data for that column.

        end % for j

    % END INNER LOOP

    % After each outer loop iteration, save workspace variables to *.mat file
        if w == 1
            save feb19_10_repdep.mat
        else
            save feb19_20_repdep.mat
        end % if w

    end % for w

    % END OUTER LOOP

    % end file
```

FILE Name: tgt_chooser.m

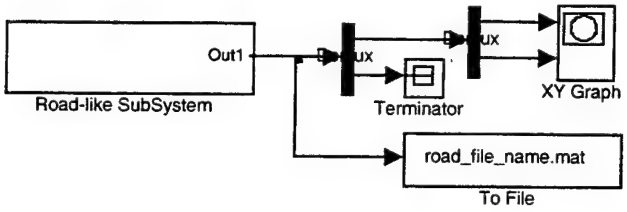
```
function [f] = tgt_chooser(k)

% function [f] = tgt_chooser(k)
% This function selects the target motion input file from 5 possible inputs.  The
% velocity information from the file is extracted into the SIMULINK model to
% provide target movement data.  Position data from the file is not used.

% This function supports MATLAB SIMULINK model "tcmts.mdl" using MATLAB
% version 5.3.0.10183 (R11).
% This was developed by CDR R. L. Mahr for the Naval Postgraduate School in partial
% fulfillment of the requirements for a Masters Degree in Aeronautical Engineering
% January 2001.  All rights reserved.

% road1 contains a straight line of movement with no target velocity changes
road1pos = k(1:2); %not used
road1vel = k(3:4);

% road2 contains a straight line of movement with random target velocity changes
road2pos = k(5:6); %not used
road2vel = k(7:8);

% road3 contains a maneuvering road for movement with no target velocity changes
road3pos = k(9:10); %not used
road3vel = k(11:12);

% road4 contains a maneuvering road for movement with random target velocity changes
road4pos = k(13:14); %not used
road4vel = k(15:16);

% offroad is a random target motion generation mode with heading and velocity changes
offroad = k(17:18);

% select is a user selectable command of the road type
select = k(19);

if select == 1
    f = road1vel;
elseif select == 2
    f = road2vel;
elseif select == 3
    f = road3vel;
elseif select == 4
    f = road4vel;
else
    f = offroad;
end % if select

% end file
```

FILE Name: true_cep.m

```
function [f] = true_cep(k)

% function [f] = true_cep(k)
% This function calculates the radial miss distance (rmd) from the missile
% to the true target position at the time of weapon 'impact'.

% This function supports MATLAB SIMULINK model "tcmts.mdl" using MATLAB
% version 5.3.0.10183 (R11).
% This was developed by CDR R. L. Mahr for the Naval Postgraduate School in partial
% fulfillment of the requirements for a Masters Degree in Aeronautical Engineering
% January 2001.  All rights reserved.

% the following inputs are received from a 1 simulation step delay.  This was done
% because the sim stop criteria are based on the first time the miss range starts
% opening up.  To get the correct rmd the last step values are used.

tgt_pos = k(1:2)*6075;        % target true position in feet from truth model
tgt_vel = k(3:4)*6075/60;     % target true velocity in feet/sec from truth model
msl_pos = k(5:6)*6075;        % missile position in feet from behavior model
msl_vel = k(7:8)*6075/60;     % missile velocity in feet/sec from behavior model
t_impact = k(9);              % calculated impact time from missile behavior model
t_minus1 = k(10)*60;          % time one sim step previous

dt = t_impact-t_minus1;       % impact can occur between sim steps, so calculate the
                              % delta time to apply to positions

% advance the last position of the missile and target using linear application of dt to
% velocity (assumes constant velocity within the simulation time step.
tgt_at_impact = tgt_pos + tgt_vel*dt;
msl_at_impact = msl_pos + msl_vel*dt;

% radial miss is the rms of the difference of the positions
d_x = (tgt_at_impact(1) - msl_at_impact(1));
d_y = (tgt_at_impact(2) - msl_at_impact(2));
radial_miss = sqrt(d_x^2+d_y^2);                    %radial miss distance in feet

% calculate range error and deflection error
alfa = atan2(msl_vel(2), msl_vel(1));
E = abs(d_y);
F = E/cos(alfa);
D = sqrt(F^2-E^2);
C = abs(d_x) - D;
B = C*sin(alfa);
A = sqrt(C^2-B^2);

RE = A;      % Range Error
DE = (B+F);% Deflection Error

f = [RE DE radial_miss];

% end file
```

APPENDIX D. TARGET TRUE MOTION MATLAB FILES

Roadtest Top Level



Brief Description:

This system creates the target motion and velocity data to be used by the TSMTS. This model must be run first to generate the data file which will be used as input to TSMTS. Once generated the data file remains static unless the model is run again. The output files must have the same name as the expected input file in the "Target Truth Subsystem" in TSMTS.

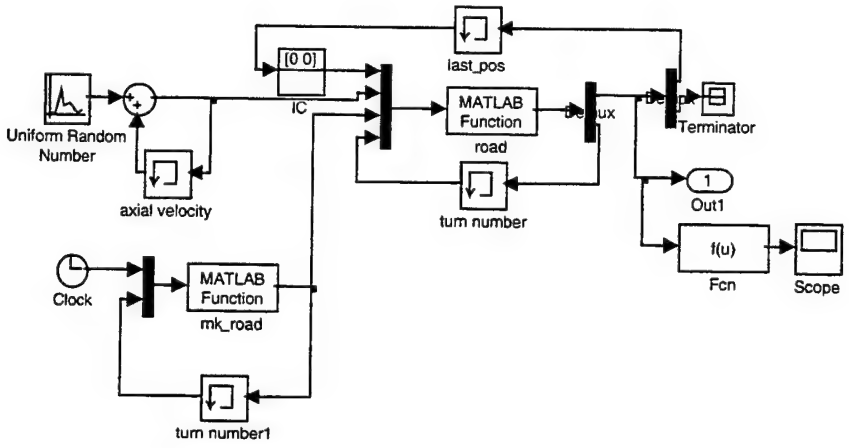
References:

None

M.file File Headers:

None

Road-like Subsystem



Brief Description:

This calls the road vector m-file and applies the target velocity to calculate the target position and velocity at each time step during the simulation. The velocity profile can be repeated by using the same seed in the Uniform Random Number block. Conversely, changing the seed will generate a new profile.

References:

None

M.file File Headers:

mk_road.m
road.m

FILE Name: mk_road.m

```
function [f] = mk_road(k)

% function [f] = mk_road(k)
% This function creates a vector representation of a road for use in predicting
% target motion.

% This function supports MATLAB SIMULINK model "tcmts.mdl" using MATLAB
% version 5.3.0.10183 (R11).
% This was developed by CDR R. L. Mahr for the Naval Postgraduate School in partial
% fulfillment of the requirements for a Masters Degree in Aeronautical Engineering
% January 2001.  All rights reserved.

t = k(1);                    % time now
road_vector = k(2:639); % static road vector

% the creation of the road significantly slows down the simulation, so it
% is created only for the first 2 iterations, at which point the memory loop
% is full, and then data can be treated as a vector.
if t < 0.002

% CONSTRUCT ROAD
    route = [0 0 0 1];          % starting point and first turn point

% create a radiused turn
    x0 = 0.5;
    y0 = 1;
    radius = 0.5;
    i = 1;

    for z = 0:.005:pi/2
        x(i) = x0+radius*sin(z);
        y(i) = y0+radius*(1-cos(z));
        route = [route  x(i) y(i)];
        i = i+1;
    end

    route = [route  5 5  10 10];        % last two turn points
% END CONSTRUCT ROAD

    f = route;
else

    f = road_vector;

end % if t

% end file
```

FILE NAME: road.m

```
function [f] = road(k)

% function [f] = road(k)
% This function generates the data for target road based motion

% This function supports MATLAB SIMULINK model "roadtest.mdl" using MATLAB
% version 5.3.0.10183 (R11).
% This was developed by CDR R. L. Mahr for the Naval Postgraduate School in partial
% fulfillment of the requirements for a Masters Degree in Aeronautical Engineering
% January 2001.  All rights reserved.

% INPUTS

last_pos = k(1:2);
v_mag = k(3)*.001;

% for road profile 1 (straight line), comment out everything from "START SECTION 1"
% through "END SECTION ONE" and uncomment "START SECTION 2" through "END
% SECTION TWO"

%%% START SECTION 1

%road_map = k(4:641);
%k(642);%k(5);
% IMPORT ROAD VECTOR DATA
%kip = length(road_map)-1;
%route = [road_map(1) road_map(2)];
%for n = 3:2:kip
%   route = [route; road_map(n) road_map(n+1)];
%end % for
% END IMPORT ROAD VECTOR DATA

%%% END SECTION 1

%%% START SECTION 2

blank = k(4);
next_turn = k(5);
route = [0 0; 10 10; 20 20; 100 100];

%%% END SECTION 2

% Behavioral Model
tn = next_turn;
dx_route = route(tn,1)-last_pos(1);
dy_route = route(tn,2)-last_pos(2);
dist2turn = sqrt(dx_route^2+dy_route^2);

if next_turn == 2 | next_turn == 3
    if dist2turn <= (300/6075)
        v_mag = (25/60)*.001;
    end
```

```
    if dist2turn <= (50/6075)
        v_mag = (5/60)*.001;
    end
end

dist_trav = v_mag;

while dist_trav >= dist2turn
    dist_trav = dist_trav - dist2turn;
    dx_route = route(tn+1,1)-route(tn,1);
    dy_route = route(tn+1,2)-route(tn,2);
    dist2turn = sqrt(dx_route^2+dy_route^2);
    tn = tn+1;
end %while
alfa = atan2(dy_route, dx_route);
rem = dist_trav*[cos(alfa) sin(alfa)];

if dist_trav < (v_mag)
    px_next = route(tn-1,1)+rem(1);
    py_next = route(tn-1,2)+rem(2);
else
    px_next = last_pos(1) + rem(1);
    py_next = last_pos(2) + rem(2);
end %if

vx = (px_next - last_pos(1))/.001;
vy = (py_next - last_pos(2))/.001;

f = [px_next py_next vx vy tn];

% end file
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX E. TABULATED RESULTS

Ordered Set of 100 runs of TCMTS

Data obtained on: **9-Feb-01**

Run Variables

Image Process Interval (secs)	0.1
Data Transmission Interval (secs)	2
TLE (ft)	0
VE (kts)	0

Results	**Road 1**	**Road 2**	**Road 3**	**Road 4**
CEP	**1.7481E-09**	**0.24296723**	**1.70970455**	**1.46719185**
Mean	**1.0308E-06**	**0.44054892**	**15.0222516**	**1.40391016**

Ordered Set of 100 runs of TCMTS

Data obtained on: **2-Feb-01**

Run Variables

Image Process Interval (secs)	0.1
Data Transmission Interval (secs)	2
TLE (ft)	50
VE (kts)	3

Results	**Road 1**	**Road 2**	**Road 3**	**Road 4**
CEP	**23.7637387**	**28.9086021**	**29.3741329**	**25.2236783**
Mean	**29.5257094**	**36.5519932**	**37.8781205**	**38.2910734**

Ordered Set of 100 runs of TCMTS

Data obtained on: **3-Feb-01**

Run Variables

Image Process Interval (secs)	5
Data Transmission Interval (secs)	2
TLE (ft)	50
VE (kts)	3

Results	**Road 1**	**Road 2**	**Road 3**	**Road 4**
CEP	**33.563263**	**30.6220651**	**37.6273979**	**34.0958172**
Mean	**40.6385012**	**37.5889294**	**80.8636281**	**111.999653**

Ordered Set of 100 runs of TCMTS

Data obtained on: **1-Feb-01**

Run Variables

Image Process Interval (secs)	10
Data Transmission Interval (secs)	2
TLE (ft)	50
VE (kts)	3

Results

	Road 1	Road 2	Road 3	Road 4
CEP	25.6265158	33.3502877	60.4265374	40.9843195
Mean	36.2434014	43.478668	132.282172	80.600683

Ordered Set of 100 runs of TCMTS

Data obtained on: **3-Feb-01**

Run Variables

Image Process Interval (secs)	10
Data Transmission Interval (secs)	2
TLE (ft)	50
VE (kts)	3

Results

	Road 1	Road 2	Road 3	Road 4
CEP	30.1772804	24.2627317	38.7429364	40.0694254
Mean	39.1267822	36.26852	101.490733	74.4036997

Ordered Set of 100 runs of TCMTS

Data obtained on: **1-Feb-01**

Run Variables

Image Process Interval (secs)	20
Data Transmission Interval (secs)	2
TLE (ft)	50
VE (kts)	3

Results

	Road 1	Road 2	Road 3	Road 4
CEP	41.8593176	43.1691023	49.1683092	61.9424346
Mean	54.0549654	53.9443717	118.877536	133.824461

Ordered Set of 100 runs of TCMTS

Data obtained on: **3-Feb-01**

Run Variables

Image Process Interval (secs)	20
Data Transmission Interval (secs)	2
TLE (ft)	50
VE (kts)	3

Results	**Road 1**	**Road 2**	**Road 3**	**Road 4**
CEP	**43.977689**	**36.4589887**	**51.960393**	**69.7612531**
Mean	**52.0487023**	**51.0610156**	**198.272208**	**133.047771**

Ordered Set of 100 runs of TCMTS

Data obtained on: **1-Feb-01**

Run Variables

Image Process Interval (secs)	30
Data Transmission Interval (secs)	2
TLE (ft)	50
VE (kts)	3

Results	**Road 1**	**Road 2**	**Road 3**	**Road 4**
CEP	**41.86367**	**57.8115301**	**66.5707189**	**62.3748189**
Mean	**60.6655229**	**72.6628549**	**193.854801**	**167.937512**

Ordered Set of 100 runs of TCMTS

Data obtained on: **4-Feb-01**

Run Variables

Image Process Interval (secs)	10
Data Transmission Interval (secs)	2
TLE (ft)	0
VE (kts)	0

Results	**Road 1**	**Road 2**	**Road 3**	**Road 4**
CEP	**2.8332E-09**	**3.48193566**	**2.1892835**	**8.75409846**
Mean	**0.00056584**	**8.94618145**	**76.11883**	**56.2434512**

Ordered Set of 100 runs of TCMTS

Data obtained on: **4-Feb-01**

Run Variables

Image Process Interval (secs)	20
Data Transmission Interval (secs)	2
TLE (ft)	0
VE (kts)	0

Results

	Road 1	Road 2	Road 3	Road 4
CEP	2.7529E-09	7.3596792	2.13284863	20.5366078
Mean	0.00149237	53.9443717	118.877536	133.824461

Ordered Set of 100 runs of TCMTS

Data obtained on: **4-Feb-01**

Run Variables

Image Process Interval (secs)	30
Data Transmission Interval (secs)	2
TLE (ft)	0
VE (kts)	0

Results

	Road 1	Road 2	Road 3	Road 4
CEP	1.0937E-08	16.5250773	2.17060899	45.4718606
Mean	0.00058768	26.0765524	147.935265	157.622913

Ordered Set of 100 runs of TCMTS

Data obtained on: **6-Feb-01**

Run Variables

Image Process Interval (secs)	10
Data Transmission Interval (secs)	2
TLE (ft)	100
VE (kts)	3

Results

	Road 1	Road 2	Road 3	Road 4
CEP	56.4859061	60.1356499	61.188039	71.8954194
Mean	64.6529598	67.5504147	96.7068262	113.067575

Ordered Set of 100 runs of TCMTS

Data obtained on: **6-Feb-01**

Run Variables

Image Process Interval (secs)	20
Data Transmission Interval (secs)	2
TLE (ft)	100
VE (kts)	3

Results	**Road 1**	**Road 2**	**Road 3**	**Road 4**
CEP	**67.5943165**	**52.0272914**	**83.0708666**	**91.8144988**
Mean	**77.3801476**	**71.0248957**	**180.718806**	**192.995009**

Ordered Set of 100 runs of TCMTS

Data obtained on: **6-Feb-01**

Run Variables

Image Process Interval (secs)	30
Data Transmission Interval (secs)	2
TLE (ft)	100
VE (kts)	3

Results	**Road 1**	**Road 2**	**Road 3**	**Road 4**
CEP	**65.4317459**	**54.8376542**	**94.3472283**	**101.298833**
Mean	**86.3127995**	**75.5132338**	**179.244944**	**234.59007**

Ordered Set of 100 runs of TCMTS

Data obtained on: **11-Feb-01**

Run Variables

Image Process Interval (secs)	10
Data Transmission Interval (secs)	2
TLE (ft)	25
VE (kts)	3

Results	**Road 1**	**Road 2**	**Road 3**	**Road 4**
CEP	**20.3648198**	**21.9045592**	**27.1320584**	**31.1277616**
Mean	**28.5450219**	**31.012832**	**71.2040595**	**74.4663101**

Ordered Set of 100 runs of TCMTS

Data obtained on: **11-Feb-01**

Run Variables

Image Process Interval (secs)	20
Data Transmission Interval (secs)	2
TLE (ft)	25
VE (kts)	3

Results

	Road 1	Road 2	Road 3	Road 4
CEP	25.1149205	29.4237154	36.7157286	41.013754
Mean	38.8881584	41.1967799	169.148881	131.473015

Ordered Set of 100 runs of TCMTS

Data obtained on: **11-Feb-01**

Run Variables

Image Process Interval (secs)	30
Data Transmission Interval (secs)	2
TLE (ft)	25
VE (kts)	3

Results

	Road 1	Road 2	Road 3	Road 4
CEP	34.9744839	38.6204936	85.3934864	53.1721376
Mean	50.4954062	60.4981814	280.888554	198.409019

Ordered Set of 100 runs of TCMTS

Data obtained on: **8-Feb-01**

Run Variables

Image Process Interval (secs)	20
Data Transmission Interval (secs)	1
TLE (ft)	50
VE (kts)	3

Results

	Road 1	Road 2	Road 3	Road 4
CEP	27.9642817	40.2279734	42.0902324	50.8605986
Mean	49.1697943	53.415456	121.314236	192.995009

Ordered Set of 100 runs of TCMTS

Data obtained on: **8-Feb-01**

Run Variables

Image Process Interval (secs)	20
Data Transmission Interval (secs)	4
TLE (ft)	50
VE (kts)	3

Results	**Road 1**	**Road 2**	**Road 3**	**Road 4**
CEP	**37.3536517**	**45.9761511**	**70.1141247**	**58.5265773**
Mean	**52.1543859**	**64.8038163**	**187.44188**	**196.342307**

Ordered Set of 100 runs of TCMTS

Data obtained on: **12-Feb-01**

Run Variables

Image Process Interval (secs)	5
Data Transmission Interval (secs)	2
TLE (ft)	50
VE (kts)	3

Results	**Random**	**Random**	**Random**	**Random**
CEP	**51.3313522**	**58.6073454**	**49.3229131**	**46.2988348**
Mean	**62.7764327**	**77.9293641**	**58.1457976**	**64.2796148**

Ordered Set of 100 runs of TCMTS

Data obtained on: **12-Feb-01**

Run Variables

Image Process Interval (secs)	10
Data Transmission Interval (secs)	2
TLE (ft)	50
VE (kts)	3

Results	**Random**	**Random**	**Random**	**Random**
CEP	**49.5986399**	**57.4402817**	**53.7067221**	**58.9750953**
Mean	**122.168544**	**66.1896415**	**78.3710853**	**97.63024**

Ordered Set of 100 runs of TCMTS

Data obtained on: **14-Feb-01**

Run Variables

Image Process Interval (secs)	20
Data Transmission Interval (secs)	2
TLE (ft)	50
VE (kts)	0

Results

	Road 1	Road 2	Road 3	Road 4
CEP	**20.72765847**	**29.1074959**	**42.8788235**	**57.6371393**
Mean	**25.63176131**	**34.4424421**	**182.387551**	**163.479541**

Ordered Set of 100 runs of TCMTS

Data obtained on: **14-Feb-01**

Run Variables

Image Process Interval (secs)	20
Data Transmission Interval (secs)	2
TLE (ft)	0
VE (kts)	3

Results

	Road 1	Road 2	Road 3	Road 4
CEP	**17.78851243**	**28.7533173**	**32.8089168**	**49.9258031**
Mean	**29.5493973**	**44.2175327**	**118.87642**	**131.179432**

Ordered Set of 1000 runs of TCMTS

Data obtained on: **16-Feb-01**

Run Variables

Image Process Interval (secs)	20
Data Transmission Interval (secs)	2
TLE (ft)	50
VE (kts)	3

Results

	Road 1	Road 2	Road 3	Road 4
CEP	**38.12163416**	**39.6366553**	**54.431163**	**57.2100272**
Mean	**47.56681657**	**52.7492246**	**167.871542**	**153.629034**

Ordered Set of 100 runs of TCMTS

Data obtained on:		**12-Feb-01**			
Run Variables					
	Image Process Interval (secs)	20			
	Data Transmission Interval (secs)	2			
	TLE (ft)	50			
	VE (kts)	3			
Results		**Random**	**Random**	**Random**	**Random**
	CEP	**78.9047024**	**70.9764165**	**84.345046**	**75.3221357**
	Mean	**104.40686**	**107.941782**	**118.723786**	**117.917135**

Ordered Set of 100 runs of TCMTS

Data obtained on:		**13-Feb-01**			
Run Variables					
	Image Process Interval (secs)	20			
	Data Transmission Interval (secs)	10			
	TLE (ft)	50			
	VE (kts)	3			
Results		**Road 1**	**Road 2**	**Road 3**	**Road 4**
	CEP	**60.0971342**	**47.6968286**	**83.2539096**	**94.845801**
	Mean	**70.5138704**	**61.3607897**	**210.469508**	**249.15098**

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Department of the Air Force, *Joint CAF and USN Operational Requirements Document (CAF 401-91-III-A) for Joint Direct Attack Munition (JDAM)* (Draft 13 Dec 99), Unclassified, 13 December 1999.

Department of the Navy, *Joint Standoff Weapon Operational Requirements Document*, Confidential, 1998.

Department of the Navy, Program Executive Officer (Tactical Aircraft Programs) PMA-201, "JSOW Unitary Program", briefing materials, undated

Driels, M., *Weaponeering*, ME-4902 classroom text, Naval Postgraduate School, 2000.

Harel, D., and Pollti, M., *Modeling Reactive Systems with Statecharts: The Statemate Approach*, I-Logix, Inc., 1999.

Hatley, D., and Pirbhai, I., *Strategies for Real-Time System Specification*, Dorset House Publishing, 1988.

Jewett, B., Captain USN, "Making Network Centric Warfare Real", briefing materials, 18 Jan 00.

Naval Postgraduate School Report, *Report on the Applicability of Current JMEM Delivery Accuracy (DA) Methodology to JDAM*, by M. Driels, September 1999.

United States Joint Forces Command, Joint Warfighting Center, *A Common Perspective*, Vol.8, No. 2, October 2000

United States Naval Academy Division of Professional Development, "What is Link 16?", http://prodevweb.prodev.usna.edu/Seanav, January 2001.

Zarchan, P., *Tactical and Strategic Missile Guidance, Second Edition,* Progress in Astronautics and Aeronautics, Vol 176, AIAA, 1997.

THIS PAGE INTENTIONALLY LEFT BLANK

SELECTED BIBLIOGRAPHY

Altshuller, G., *The Innovation Algorithm*, Technical Innovation Center, Inc., 1999.

Naval Postgraduate School Report, *Report on the Applicability of Current JMEM Delivery Accuracy (DA) Methodology to JDAM*, by M. Driels, September 1999.

Shneydor, N. A., *Missile Guidance and Pursuit*, Horwood Series in Engineering Science, Horwood Publishing Ltd., Chichester, 1998.

Swee, J., *Missile Terminal Guidance and Control Against Evasive Targets,* Thesis, Naval Postgraduate School, March 2000.

THIS PAGE INTENTIONALLY LEFT BLANK

GLOSSARY

2-D	Two dimensional.
3-D	Three-dimensional.
3-DOF	Three degree of freedom, used to describe the fidelity of a kinematic model.
A-6	Two-seat all weather U.S. Navy attack aircraft, retired from the Navy inventory in 1995.
AIWS	Advanced Interdiction Weapon System, an early name for JSOW.
B/N	Bombardier/Navigator, the right seat crewman in an A-6.
C3I	Command, Control, Communications and Intelligence.
CAIV	Cost as an Independent Variable.
CAP	Combat Air Patrol.
CEP	Circular Error Probable, the most common measure of measure of weapon miss distance. The CEP is a circle centered on the desired mean point of impact with a radius such that 50% of all weapons delivered lie within the circle.
DE	Deflection Error, the miss distance in the deflection direction.
DEP	Deflection Error Probable, the distance to one of a pair of lines perpendicular to the range direction and spaced so 50% of all weapons impact between them.
DFC	Data Fusion Cell, proposed high-speed ground station to support the TSMTS.
DMPI	Desired mean point of impact, the point at which the weapon, or center of a pattern of weapons, is aimed.
DP/FS	Data Processing and Fusion System, the proposed combination of DPS and DFC.
DPS	Data Processing Station, proposed high speed ground station to support the TSMTS
DR	Dead reckoning, a technique of tracking position based on estimates of velocity and other factors.
DTI	Data Transmission Interval, the periodic interval between data transmissions.
DTS	Data Transmission System proposed ground and air transmitters and receivers to support the TSMTS.
F/A-18	Single seat U.S. Navy Strike/Fighter aircraft.
GPS	Global Positioning System, a network of ground stations, non-geosynchronous satellites, and receivers that provide highly accurate positioning at or above the earth's surface.

IPI	Image Processing Interval, the time between receiving an image, and when the data from that image is available in the TSMTS.
JDAM	Joint Direct Attack Munition (GBU-29,30,31,32), Navy and Air Force inventory weapon kit that provides precision accuracy to inventory unguided conventional warheads.
JMEM	Joint Munition Effectiveness Manuals, a publication of the JTCG that provide detailed information for weaponeering of targets.
JSOW	Joint Standoff Weapon (AGM-154 A/B/C), Navy and Air Force inventory long-range glide weapon capable of delivering a variety of submunitions or unitary payloads.
JSTARS	Joint Surveillance and Target Attack Reconnaissance System.
JTCG	Joint Technical Coordinating Group, chartered by OSD to collect, evaluate and disseminate target vulnerability information.
KBPS	Kilobits per second.
Kt	Knot, one nautical mile per hour.
KPP	Key Performance Parameter.
MIDS	Multi-functional Information Distribution System.
MITL	Man-in-the-Loop, used to refer to systems which allow operator input.
NIMA	National Imagery and Mapping Agency.
NM	Nautical mile.
NPS	Naval Postgraduate School, a top ten institution.
ORD	Operational Requirements Document, the formal document which describes the need for and capabilities required of any military weapon system.
RE	Range Error, the miss distance in the deflection direction.
REP	Range Error Probable the distance to one of a pair of lines perpendicular to the deflection direction and spaced so 50% of all weapons impact between them.
RMD	Radial miss distance, the magnitude of the vector connecting the DMPI with the point of actual impact or center of the pattern.
SLAM	Standoff Land Attack Missile, Navy inventory weapon derived from the Harpoon missile that is capable of attacking land based targets using preplanned or MITL attacks.
SLAM-ER	SLAM (Enhanced Response), Navy inventory weapon, improved version of the SLAM that incorporates range and accuracy improvements.
SSPD	Single Sortie Probability of Damage, the probability of inflicting a specific level of damage on one sortie.

SWMP	Strike Warfare Master Plan.
TLE	Target Location Error, the difference between the location of the target in the physical world, and the reported position from the intelligence system.
TSMT	Time Sensitive Moving Target, a TST that is, or is capable of, moving.
TSMTS	Time Sensitive Moving Target System.
TST	Time Sensitive Target, those targets requiring immediate response because they pose (or will soon pose) a clear and present danger to friendly forces or are highly lucrative, fleeting targets of opportunity.
TVE	Target Velocity Error, the difference between the velocity of the target in the physical world, and the reported position from the intelligence system.
WGS-84	World Geodetic Survey 1984, the reference system used by most GPS systems.
UHF	Ultra-high Frequency.
VMF	Variable Message Format.
VPF	Vector Product Format.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
   8725 John J. Kingman Road, Suite 0944
   Ft. Belvoir, VA 22060-6218

2. Dudley Knox Library 2
   Naval Postgraduate School
   411 Dyer Road
   Monterey, CA 93943-5101

3. Professor Max F. Platzer, Code AA/Pl 1
   Chairman, Department of Aeronautics and Astronautics
   Naval Postgraduate School
   Monterey, CA 93943-5000

4. Professor Russell W. Duren, Code AA/Dr 1
   Department of Aeronautics and Astronautics
   Naval Postgraduate School
   Monterey, CA 93943-5000

5. Professor Morris Driels, Code ME/DR 1
   Department of Mechanical Engineering
   Naval Postgraduate School
   Monterey, CA 93943-5000

6. CAPT R. Wirt, PEO(W), PMA-201 1
   Building 2272
   47123 Buse Rd.
   Patuxent River, MD 20670

7. Mr. J. Tom Glass 1
   22685 Three Notch Rd.
   California, MD 20619

8. Mr. Daniel McInnis 1
   AAC/ENM
   101 W. Eglin Blvd, Suite 384
   Eglin AFB, FL 32542

9. Ms. Carolyn Holland 1
   AAC/ENMS
   101 W. Eglin Blvd, Suite 384
   Eglin AFB, FL 32542

10. Mr. Bill Swift 1
AAC/ENMS
101 W. Eglin Blvd, Suite 384
Eglin AFB, FL 32542

11. Mr. William Tonkin 1
NAWC-WD 4J6000D
1 Administration Circle
China Lake, CA 93555

12. Mr. Art Rosenbaum 1
OSU Field Office
503 W. VanMatre
Eglin AFB, FL 32542

13. CDR Randolph L. Mahr, USN 1
5516 Ferndale St
Springfield, VA 22151